THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
l'Université Toulouse III - Paul Sabatier

Présentée et soutenue par Imen KETATA
Le 11 Janvier 2012

Titre : Méthode de découverte de sources de données tenant compte de la sémantique en environnement de grille de données

---

ED MITT : Domaine STIC : Réseaux Télécoms, Systèmes et Architecture

Discipline ou spécialité : Informatique

Unité de recherche : Institut de Recherche en Informatique de Toulouse

Directeur de Thèse :

Franck MORVAN
Professeur à l'Université Paul Sabatier Toulouse III

Rapporteurs :

Chirine GHEDIRA-GUEGAN Professeur à l'Université Jean Moulin Lyon 3
Samir TATA Professeur à TELECOM SudParis

Autres membres du jury :

Abdelkader HAMEURLAIN Professeur à l'Université Paul Sabatier Toulouse III	Examinateur
Nouredine MELAB Professeur à l'Université Lille I	Examinateur
Riad MOKADEM Maître de Conférences à l'Université Paul Sabatier Toulouse III	Encadrant

A ma famille,

A ceux que j'aime.

Remerciements

J’adresse mes remerciements tout d’abord à Messieurs Luis FARINAS DEL CERRO et Michel DAYDE, respectivement ancien et actuel Directeur de l’Institut de Recherche en Informatique de Toulouse (IRIT), pour m’avoir accueillie au sein du laboratoire.

Je tiens à remercier Madame le Professeur Chirine GHEDIRA-GUEGAN pour l’intérêt qu’elle a accordé à mon travail en acceptant d’en être rapporteur. Je remercie également Monsieur le Professeur Samir TATA d’avoir accepté de rapporter sur mon travail. Les différentes remarques de Madame GHEDIRA-GUEGAN et Monsieur TATA ont contribué à améliorer la qualité de ce manuscrit. Je tiens à remercier aussi Monsieur le Professeur Nouredine MELAB d’avoir consacré du temps à l’examen de ce document en acceptant de participer au jury.

J’adresse également mes vifs remerciements à Monsieur le Professeur Abdelkader HAMEURLAIN de m’avoir accueillie au sein de son équipe Pyramide. Je le remercie pour ses précieux conseils qui m’ont poussé toujours à m’améliorer.

Toute ma reconnaissance s’adresse à Monsieur le Professeur Franck MORVAN, mon directeur de recherche pour sa disponibilité dans le suivi de ce travail. Je le remercie pour sa rigueur, sa patience et ses précieux conseils tant sur le plan scientifique que humain. Je le remercie de trouver toujours les bons mots pour m’encourager dans les moments difficiles ce qui m’a permis de réussir cette thèse.

Je remercie beaucoup Monsieur Riad MOKADEM, Maître de Conférences, de son encadrement et tout le temps qu’il m’a accordé pour suivre le moindre détail dans ce travail. Son aide et ses remarques constructives m’ont considérablement aidé à améliorer ce mémoire.

J’associe à mes remerciements l’ensemble du personnel du laboratoire IRIT ainsi que de l’Université Paul Sabatier.

J’adresse un grand merci à tous mes amis avec qui j’ai partagé mes moments de joie et qui m’ont beaucoup soutenu pendant les périodes difficiles aussi. Commençant par ceux de l’équipe Pyramide, les aniciens : Nadhem, Raddad et Mahmoud, et les nouveaux Deniz et Igor. Je remercie également mes amis des autres équipes de l’IRIT en particulier l’équipe SIG : Arlind (ou Kopss) avec toutes ses bêtises, Cyrilopédia, Hamdi, Amjed, Firas, Ines, Faten, Dana, Madalina, la liste est encore longue. Je tiens à remercier aussi mes amis des autres universités, particulièrement Fatma, pour son soutien continu et d’être ma confidente quand j’en avais besoin. Merci à Céline d’être toujours là avec ses soutiens. Un grand merci à Brahim et Laura, pour leur gentillesse, patience et tout ce qu’on a partagé! Une pensée à Mohamed, Irene, Riadh, Ismail et Emna pour leur soutien malgré la distance!

Le meilleur pour la fin, toute ma gratitude à ma famille, mon papa Mohamed, ma maman Nesria, mes deux sœurs Sonia et Nesrine et mon petit frère Karim qui ont toujours cru en moi et m’ont soutenu jusqu’au bout malgré la distance. J’oubli pas de remercier mes deux beaux-frères Mahdi et Noômen et mon petit neveu Mohamed. Sans cette aimable famille je ne réussirai jamais cette thèse.

Résumé

Actuellement, les applications grilles de données recherchent à partager un nombre gigantesque de sources de données en un environnement instable où une source de données peut à tout moment joindre ou quitter le système. Ces sources de données sont fortement hétérogènes parce qu'elles sont développées indépendamment, éparpillées géographiquement et gérées de manière autonome. Dans cet environnement, la découverte efficace des sources de données pertinentes pour l'exécution de requêtes est un problème complexe en raison de l'hétérogénéité des sources, de la grande échelle et de l'instabilité du système.

Les premiers travaux sur la découverte de sources de données se sont basés sur une recherche par mots clés. Ces premières solutions ne sont pas satisfaisantes puisqu'elles ne tiennent pas compte des problèmes de l'hétérogénéité sémantique des sources de données. Ainsi, la communauté a proposé d'autres solutions pour prendre en compte les aspects sémantiques. Une première solution consiste à utiliser un schéma global ou une ontologie globale. Cependant, la conception d'un tel schéma ou d'une telle ontologie est une tâche complexe à cause du nombre de sources de données. D'autres solutions on été proposées en proposant des correspondances entre les schémas des sources de données ou en s'appuyant sur des ontologies de domaine et en établissant des relations de 'mapping' entre ces dernières. Toutes ces solutions imposent une topologie fixe soit pour les correspondances soit pour les relations de 'mapping'. Cependant, la définition de relations de 'mapping' entre ontologies de domaine est une tâche ardue et imposer une topologie fixe est un inconvénient majeur.

Dans cette perspective, nous proposons dans cette thèse une méthode de découverte de sources de données prenant en compte les problèmes liés à l'hétérogénéité sémantique en environnement instable et à grande échelle. Pour cela, nous associons une Organisation Virtuelle (OV) et une ontologie de domaine à chaque domaine et nous nous basons sur les relations de 'mappings' existantes entre ces ontologies. Nous n'imposons aucune hypothèse

sur la topologie des relations de ‘mapping’ mis à part que le graphe qu’elles forment soit connexe. Nous définissons un système d’adressage permettant un accès permanent de n’importe quelle OV_i vers une autre OV_j malgré la dynamicité des pairs (avec $i \neq j$). Nous présentons également une méthode de maintenance dite ‘paresseuse’ afin de limiter le nombre des messages nécessaires à la maintenance du système d’adressage lors de la connexion ou de la déconnexion de pairs.

Pour étudier la faisabilité ainsi que la viabilité de nos propositions, nous effectuons une évaluation des performances.

Mots-clés

Grille de Données, Sources de Données Réparties à Grande Échelle, Découverte de Ressources, Évaluation de Performances.

Résumé en anglais

Nowadays, data grid applications look to share a huge number of data sources in an unstable environment where a data source may join or leave the system at any time. These data sources are highly heterogeneous because they are independently developed and managed and geographically scattered. In this environment, efficient discovery of relevant data sources for query execution is a complex problem due to the source heterogeneity, large scale environment and system instability.

First works on data source discovery are based on a keyword search. These initial solutions are not sufficient because they do not take into account problem of semantic heterogeneity of data sources. Thus, the community has proposed other solutions to consider semantic aspects. A first solution consists in using a global schema or global ontology. However, the conception of such scheme or such ontology is a complex task due to the number of data sources. Other solutions have been proposed providing mappings between data source schemas or based on domain ontologies and establishing mapping relations between them. All these solutions impose a fixed topology for connections as well as mapping relationships. However, the definition of mapping relations between domain ontologies is a difficult task and imposing a fixed topology is a major inconvenience.

In this perspective, we propose in this thesis a method for discovering data sources taking into account semantic heterogeneity problems in unstable and large scale environment. For that purpose, we associate a Virtual Organisation (VO) and a domain ontology to each domain and we rely on relationship mappings between existing ontologies. We do not impose any hypothesis on the relationship mapping topology, except that they form connected graph. We define an addressing system for permanent access from any OVi to another OVj despite peers' dynamicity (with $i \neq j$). We also present a method of maintenance called 'lazy' to limit

the number of messages required to maintain the addressing system during the connection or disconnection of peers.

To study the feasibility as well as the viability of our proposals, we make a performance evaluation.

Mots-clés en anglais

Data Grid, Data Sources Distributed in Large Scale, Resource Discovery, Performance Evaluation.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1 : Introduction

Chapitre 1 :

Introduction

1. Contexte et motivation
2. Problématique et objectifs
3. Contribution
4. Organisation du manuscrit

Introduction

1. Contexte et motivation

Actuellement le nombre de sources de données accessibles via Internet ne cesse de croître. Cette croissance a donné naissance à de nouvelles applications référençant des données éparpillées sur des sources de données autonomes, hétérogènes et dynamiques. À tout instant, une source de données peut se connecter ou se déconnecter du réseau. Développer des applications dans un environnement à grande échelle et dynamique est devenu un réel challenge. Un exemple d'application que nous utiliserons souvent dans cette thèse est la recherche dans le domaine biomédical. En effet, un grand nombre de sources de données sont publiquement accessibles via le net et concernent une variété de données comme les expressions et les séquences des gênes, les fonctions et structures moléculaires, les caractéristiques des maladies, etc. Ces sources de données sont fortement hétérogènes puisqu'elles sont développées indépendamment les unes des autres (i.e., des pages web, des fichiers texte, des programmes, des SGBDs…). Nous citons, à titre d'exemple les sources de données biomédicales, comme : GenBank et SwissProt. Les sources de données sont également dynamiques puisqu'à tout moment elles peuvent se connecter ou se déconnecter du réseau. Enfin, elles sont très nombreuses et elles stockent des volumes de données gigantesques. Dans un tel contexte, la recherche des sources de données pertinentes pour une requête utilisateur est un vrai défi. On parle généralement, dans la littérature, de la découverte de ressources.

2. Problématique et objectifs

La découverte de ressources a déjà fait l'objet de nombreuses publications [Gnu00, Cla00, Sto01, Can01, Bos91, Kaz01, Nap01, Rat01, Row01, Zha01, Sen02, Cha03, Lia06, Pac07, Tal07a, Tru07, Mes08, Ran08]. Les ressources à découvrir sont soit relatives à des ressources de calcul, comme par exemple la découverte d'une machine avec un processeur Intel Pentium T5200 avec au

moins 2Go de mémoire vive libre, soit relative à une source de données, comme une base de données, un fichier XML, un fichier texte, une page web, un programme, etc. Une des différences fondamentales entre les deux types de découverte est que pour une ressource de calcul, on recherche une ressource quelconque vérifiant certaines contraintes. Ainsi, toute ressource vérifiant les contraintes exprimées peut être utilisée. Concernant la découverte de sources de données, on recherche une source précise nécessaire à l'évaluation d'une requête. Sans cette source de données, l'évaluation de la requête n'est pas possible. Dans cette thèse, nous nous intéressons uniquement à la découverte de ressources de type sources de données.

Les chercheurs se sont d'abord concentrés sur la découverte de sources de données en se basant sur les mots clés, tel que PeerDB [Sio05]. Par exemple, on recherche les sources de données relatives au mot clé docteur. Vers la fin des années 90, les premières solutions [Kau07, Mol08, Yu03] se basaient sur un serveur central qui stockait la description de toutes les sources de données. Ces solutions se sont vite avérées limitées. En effet, avec le nombre croissant de sources de données et d'utilisateurs, le serveur central devenait un goulet d'étranglement pour le système. Il limitait ainsi les performances du processus de découverte. Les chercheurs se sont ensuite concentrés à décentraliser le contrôle. Ils se sont donc appuyés sur les systèmes pair-à-pair. Les méthodes de recherche d'une source de données et de mises à jour dues à la dynamicité des sources de données (i.e. une source peut être ajoutée ou enlevée du système à tout moment) et des pairs, dépendent fortement de la topologie des systèmes pairs-à-pair sous-jacents. Dans les systèmes non-structurés, les coûts de mises à jour sont faibles grâce à un haut degré d'autonomie alors que les méthodes de découverte peuvent saturer le réseau à cause du mécanisme de diffusion utilisé dans ces systèmes. De plus, ces méthodes ne sont pas fiables à cause du nombre de sauts limité entre les pairs. En effet, une source de données peut ne pas être trouvée alors qu'elle existe dans le réseau. A l'inverse, les méthodes de découverte dans les systèmes pair-à-pair structurés sont fiables et relativement efficaces grâce à leur système d'indexation. Cependant, les coûts de mises à

jour dus à la dynamicité des pairs sont importants. Quant aux systèmes hybrides, ils combinent les avantages et les inconvénients des systèmes structurés et non-structurés. Cependant, quelle que soit la méthode proposée (non-structurée, structurée ou hybride), elles ne sont pas satisfaisantes car elles ne tiennent pas compte des problèmes de l'hétérogénéité structurelle et sémantique des sources de données [Cru04a, Zhu04b, Liu05, Cas04, Ket10]. Plusieurs travaux de recherche se sont intéressés à la prise en compte de ces problèmes dans la découverte de source de données. Les premiers travaux se sont principalement intéressés à l'hétérogénéité liée à la structure des schémas (e.g, différence de format, structure, complexité des schémas de données) [Cru04a, Zhu04, Liu05, Cas04, Rey03, Sar04, Sio05]. Par la suite, plusieurs travaux ont cherché à introduire la sémantique dans la découverte. En effet, du fait que les sources de données sont développées de manière autonome, nous sommes confrontés à des problèmes de :

(i) synonymie où des données sémantiquement similaires sont représentées sous différents termes dans plusieurs sources de données ;
(ii) polysémie où des données sémantiquement différentes sont représentées par le même terme.

Dans ce contexte, trois approches ont été proposées [Ket11a]: (i) L'approche basée sur la correspondance entre les noms utilisés dans les schémas des sources de données [Sio05], [Ban04] et [Li05]. Un inconvénient majeur de cette approche est la maintenance des correspondances dans un environnement fortement dynamique, (ii) L'utilisation d'un schéma global ou d'une ontologie comme schéma pivot [Cru04], [Akb04] et [Haa04]. Cependant, la conception d'un schéma global ou d'une ontologie globale demeure une tâche complexe à réaliser devant le nombre gigantesque de sources de données dans un domaine tel que le domaine biomédical et (iii) l'usage de différentes ontologies de domaine [Alk08], [Hei04], [Zhu04a] et [Ket11b]. Dans cette dernière approche, à chaque domaine est associé une ontologie de domaine. Cette approche est la plus prometteuse puisqu'elle conserve l'autonomie de chaque domaine. Entre les différentes ontologies, des relations de 'mappings'

définissent les liens de correspondances entre ces dernières. Néanmoins, toutes les méthodes de découverte de sources de données proposées dans le cadre de cette approche imposent une topologie pour les relations de ‘mapping’. La définition de relations de ‘mapping’ entre ontologies de domaine est une tâche ardue. Ainsi, imposer une topologie fixe est un inconvénient majeur. En effet, il existe de manière disponible sur Internet des ontologies de domaine ainsi que des relations de ‘mapping’ entre ces ontologies. La topologie entre ces relations de ‘mapping’ est quelconque.

3. Contribution

Dans le cadre de la découverte de sources de données, l’objectif de ce document est triple : (i) analyser les principales méthodes de découverte de sources de données proposées dans la littérature ; (ii) mettre en évidence les différentes caractéristiques que doit posséder une méthode de découverte de ressources ; (iii) proposer et évaluer une méthode de découverte de ressources en se basant sur ces caractéristiques.

La méthode de découverte de sources de données proposée prend en compte les différentes caractéristiques d’une grille de données qui sont l’hétérogénéité, la dynamicité et la grande échelle. Une organisation virtuelle est associée à un ensemble de pairs stockant des sources de données utilisant une même ontologie de domaine comme schéma pivot. La découverte au sein d’une OV est effectuée en utilisant une table de hachage distribuée (THD) classique. Ensuite, pour découvrir une source de données entre les organisations virtuelles nous proposons un système d’adressage permettant un accès permanent de n’importe quelle OV vers une autre. L’avantage de notre système d’adressage est qu’il n’est pas spécifique à une topologie de ‘mappnig’ existant entre les différentes ontologies de domaines utilisées comme schémas pivots. Il s’adapte à n’importe qu’elle topologie. Notre seule hypothèse est que le graphe non-orienté que forment les ontologies et leurs ‘mappings’ soit connexe. Le système d’adressage proposé évite que tous les pairs appartenant à OV_i soient connectés vers un même pair d’une OV_j s’il existe un lien de

‘mapping’ entre les ontologies O_i et O_j utilisées comme schémas pivots respectivement par les OV_i et OV_j. Le système permet d’éviter d’une part qu’un pair forme un goulet d’étranglement et d’autre part que la déconnexion d’un pair volontaire ou non coupe l’accès entre une OV_i et une OV_j.

Pour mettre à jour le système d’adressage après la déconnexion d’un pair nous adoptons une maintenance dite « paresseuse » afin de réduire le nombre de messages nécessaires à la maintenance du système pour éviter au maximum l’effet ‘Churn’. Cette maintenance est effectuée au moment de la découverte de sources de données et non pas quand le pair se déconnecte.

4. Organisation du manuscrit

L’organisation de ce document est la suivante.

Le chapitre 2 a pour but de proposer une synthèse des différentes méthodes de découverte de ressources proposées dans la littérature. Nous décrivons d’abord les méthodes de découverte de sources de données par mots clés. Ensuite, nous proposons un ensemble de critères permettant de les comparer et nous discutons les avantages et les inconvénients des différentes approches proposées en fonction des critères définis. Puis, nous présentons les méthodes de découverte prenant en compte les aspects sémantiques. Enfin, nous effectuons une analyse critique pour définir l’ensemble des caractéristiques que doit posséder une méthode de découverte de sources de données en environnement à grande échelle et dynamique.

Le chapitre 3 décrit une proposition d’une méthode de découverte de ressources dans un environnement à grande échelle et dynamique. Nous commençons par définir l’environnement et l’architecture de notre système. Ensuite, nous décrivons notre méthode de découverte de sources de données tout en prenant en compte l’hétérogénéité sémantique valide pour une topologie de ‘mapping’ entre les VOs des domaines quelconque. Cette description concerne deux parties : (i) une première partie pour le système

d’adressage du processus de découverte de sources de données et (ii) une deuxième partie pour la maintenance du système.

Le chapitre 4 est consacré à l’évaluation des performances de notre proposition. Pour mesurer la viabilité de notre méthode, nous réalisons une étude qualitative et quantitative. En raison du nombre limité de pairs connectés à un réseau à grande échelle auquel nous avons accès, nous évaluons, par, simulation les performances de notre méthode. L’évaluation des performances de la méthode proposée est effectuée à travers une comparaison de celle-ci avec trois autres méthodes de découverte de sources de données dans la littérature.

Enfin, le chapitre 5 établit une synthèse du travail proposé au sein de ce document en le situant par rapport à la littérature. Nous proposons également dans ce dernier chapitre les problèmes ouverts et quelques perspectives pouvant être envisagées comme travaux futurs.

Chapitre 2 : Découverte de sources de données : État de l’art

Chapitre 2 :
Découverte de sources de données :
État de l'art

1. Introduction
2. Découverte de sources de données basée sur une recherche par mots clés
   - 2.1. Approches centralisées
   - 2.2. Approches basées sur les systèmes pair-à-pair
     - 2.2.1. Approches basées sur les systèmes pair-à-pair non-structurés
     - 2.2.2. Approches basées sur les systèmes pair-à-pair structurés
     - 2.2.3. Approches basées sur les systèmes pair-à-pair hybrides
   - 2.3. Comparaison
     - 2.3.1. Critères de comparaison
     - 2.3.2. Discussion (Comparaison des trois types de systèmes basés sur la recherche par mot clés)
3. Découverte de sources de données tenant compte de la sémantique
   - 3.1. Approches basées sur des correspondances entre mots clés
   - 3.2. Approches utilisant une ou plusieurs ontologies ou schémas
     - 3.2.1. Approche utilisant une ontologie globale ou schéma global
     - 3.2.2. Approche utilisant plusieurs ontologies de domaines ou schémas
     - 3.2.3. Discussion
4. Conclusion

Découverte de sources de données : État de l'art

1. Introduction

La découverte de ressource attire de plus en plus l'attention des chercheurs dans le domaine de gestion des données vu l'importance de ce processus dans l'évaluation d'une requête utilisateur au sein d'un environnement dynamique et à grande échelle. Dans ce contexte, découvrir les ressources revient à découvrir des métadonnées décrivant ces ressources. Les métadonnées d'une ressource de données sont des informations qui renseignent sur l'existence et la description d'une source de données. Beaucoup de chercheurs se sont intéressés au processus de découverte de ressources. Ces ressources peuvent être classées sous deux types : les ressource de calculs (e.g., CPU, E / S, bande passante réseau) et les sources de données (e.g., une base de données, un fichier XML, un fichier texte, une page web, un programme, etc). Dans cette thèse, nous nous intéressons uniquement aux ressources du type sources de données.

Découvrir les sources de données de manière efficace au sein d'un environnement dynamique et à grande échelle constitue un grand défi. Cette découverte de sources de données est de plus en plus complexe à cause du nombre important de sources de données hétérogènes via le net qui ne cesse d'augmenter. En plus, ces sources de données sont de plus en plus dynamiques. En effet, une source de données peut rejoindre ou quitter le système à tout moment.

Les sources de données peuvent être sous plusieurs formes : des pages web, des fichiers XML, des fichiers texte, des programmes, etc. Chaque source de données est décrite par un ensemble de métadonnées. À titre d'exemple, une source de données peut avoir comme métadonnées utiles son adresse IP de telle

sorte que nous puissions la retrouver ou encore son profil comprenant sa taille qui sera utilisé lors du processus d'optimisation.

Dans ce chapitre, nous présentons une synthèse de quelques travaux de recherche concernant des méthodes de découverte de sources de données efficaces. Nous présentons également une description des avantages et des inconvénients de chacune d'elles. Tout d'abord, nous détaillons les mécanismes de découverte de sources de données employés dans la majorité des travaux s'appuyant sur une recherche par mots clés. Nous soulignons leurs limites face aux besoins exigés. Parmi ces exigences, nous pouvons citer les problèmes liés à la polysémie, la synonymie, les abréviations, etc. Résoudre ces problèmes revient à prendre en considération l'aspect sémantique dans le processus de découverte de sources de données. Ainsi, nous présentons, dans la section suivante, les principales méthodes existantes de découverte de sources de données.

2. Découverte de sources de données basée sur une recherche par mots clés

2.1. Approches centralisées

Vers la fin des années 90 et suite au succès qu'a eu le réseau Internet avec tous ses services fournis aux utilisateurs, des logiciels de partage de données à grande échelle ont vu le jour. Nous citerons comme exemple le système Napster [Nap01] pour le partage de fichiers musicaux. Le partage de données est basé sur une architecture centralisée [Kau07, Mol08, Yu03]; chaque nœud[1] ou pair du réseau déclare les fichiers qu'il souhaite partager à un serveur central. Le serveur conserve de son coté une adresse IP et un numéro de port pour chaque fichier. Cependant, ces services centralisés se sont vite avérés limités face aux besoins des internautes via les logiciels de partage à grande échelle. En particulier, un serveur central qui détient la responsabilité de contrôler tout le système a une capacité limitée de tenir un nombre important

[1] Nœud et pair ont le même sens dans tous les chapitres.

de pairs. Il forme naturellement un goulet d'étranglement, ce qui limite en conséquence, le passage à l'échelle. S'ajoute à cela, sa vulnérabilité par rapport à la panne du serveur central qui paralyse automatiquement tout le système qui devient inutilisable.

En réponse à ces limites, les chercheurs se sont penchés vers des services de découverte de sources de données dans des systèmes hiérarchiques [Elm05, Ram06, Ham09] et décentralisés comme le système KaZaA [Kaz01, Kaz03]. Les systèmes décentralisés ont prouvé leur efficacité par rapport au passage à la grande l'échelle. Pour expliciter d'avantage ces systèmes nous citerons l'exemple de KaZaA [Kaz01, Kaz03]. KaZaA est basée sur une architecture pair-à-pair décentralisée. Les pairs (internautes) sont reliés entre eux de façon directe sans le recours à un serveur central. Cette décentralisation rend les pairs plus indépendants par rapport à l'architecture client-serveur permettant à tous les pairs de jouer, à la fois, le rôle du client et du serveur.

De nos jours, les systèmes dits pair-à-pair, connu aussi sous l'abréviation P2P, connaissent de plus en plus de succès, notamment pour le partage de données [Sar03]. Le partage de fichiers dans ces systèmes pair-à-pair décentralisés assure une meilleure disponibilité des données grâce au fait que les données peuvent être répliquées (l'exemple le plus réputé étant celui relatif au partage de fichiers musicaux). L'utilisation de tels systèmes permet également de décharger les pairs en termes de quantité de requêtes utilisateur traitées vu qu'un pair ne se connecte pas à un serveur unique mais plutôt à plusieurs. Par la suite, une déconnexion (e.g., panne) de pair ne bloque pas le processus de recherche d'une donnée. Par conséquent, une telle architecture permet de pallier les inconvénients de la dynamicité des pairs (connexion / déconnexion) et d'avoir ainsi un système plus robuste avec une capacité de passer à l'échelle plus importante.

2.2. Approches basées sur les systèmes pair-à-pair

Les systèmes basés sur une architecture décentralisée ont été classés en trois types : (i) les systèmes pair-à-pair non-structurés (e.g., Gnutella [Gnu00, Cha03]), (ii) les systèmes pair-à-pair structurés (e.g., Chord [Sto01], CAN [Can, Bos91, Rat01], Pastry [Row01], Freenet [Cla00], Tapestry [Zha01]) et (iii) les systèmes hybrides (e.g., FastTrack [Sen02, Lia06]). Dans ces systèmes, la méthode de découverte de sources de données dépend du type de système pair-à-pair sous-jacent. Nous décrivons ces méthodes de découverte de sources de données de façon plus détaillée dans ce qui suit.

2.2.1. Approches basées sur les systèmes pair-à-pair non-structurés

Dans les systèmes pair-à-pair non-structurés [Iam01, Kal02, Iam04, Abd05, Tal05, Jea08], le processus de gestion et de contrôle au niveau des pairs n'est pas centralisé et ne respecte aucune topologie particulière de réseau. Tous les pairs jouent des rôles équivalents. Chaque pair stocke et gère les métadonnées de ses sources locales indépendamment des autres pairs. Il dispose ainsi d'un degré d'autonomie élevé. Pour le routage d'une requête de découverte de source de données, un pair adopte le mécanisme de diffusion. Ce processus doit cependant respecter une contrainte qui consiste à respecter le nombre maximum de sauts que la requête peut effectuer. Ce nombre de sauts est appelé TTL (Time-To-Live) dans la littérature. Ce paramètre équivaut à la longueur du chemin en termes de nombre de pairs visités. Le processus de routage est le suivant : l'envoi d'une requête de découverte se fait initialement par un pair initiateur (N_i). Après avoir reçu une requête utilisateur, ce pair N_i envoie une requête de découverte vers tous ses voisins de façon identique. Chaque pair se trouvant à une distance inférieure ou égale au TTL recevra la requête de découverte. La recherche se fait alors de façon locale au niveau de chaque pair recevant cette requête. Puis, ce pair envoie la même requête de découverte à ses voisins tout en diminuant la valeur de TTL de 1 à chaque saut. Quand le TTL atteint zéro, le pair concerné détruira la requête de découverte.

Ce type d'approche pair-à-pair non-structuré offre un degré d'autonomie de pairs important et évite la centralisation. Cependant, nous pouvons constater en contre partie que seuls les pairs situés à une distance inférieure au TTL recevront la requête de découverte. Ainsi, les autres pairs ne seront pas contactés même s'ils ont le résultat ou une partie du résultat. En conséquence, une requête de découverte de sources peut avoir comme résultat une réponse incomplète. Ceci s'appelle dans la littérature les fausses réponses ou réponses erronées. Un autre inconvénient majeur provient du système de propagation de la requête. En effet, un processus de diffusion à tous les voisins par inondation est utilisé dans de tels systèmes ce qui peut amener à la saturation du réseau. Ceci est bien évidemment un obstacle devant le passage à l'échelle.

Beaucoup de travaux ont été menés dans le but de combler cette limite [Pap05]. Il y a ceux qui ont pensé à une solution basée sur un mécanisme d'inondation contrôlée appelé '*Dynamic Querying*' [Fis]. En se basant sur ce mécanisme, le pair initiateur (qui a reçu la requête de découverte) se charge de contrôler la diffusion des requêtes de découverte en les envoyant à une partie des voisins choisis aléatoirement et non pas à la totalité [Kal02]. Ainsi, cette solution garantit une diminution au niveau du nombre des messages échangés. Néanmoins, le fait d'envoyer la requête de découverte de sources de données à uniquement une partie des voisins augmente la probabilité d'avoir des fausses réponses. [Fis] Etend cette solution en employant un petit TTL et au cas où ce processus n'aboutit pas à des résultats satisfaisants, le pair initiateur reprend la propagation de la requête de découverte mais vers un autre ensemble de voisins et avec un TTL plus important si nécessaire.

Plusieurs autres techniques ont été proposées dans cette même finalité afin de réduire le nombre de messages [Tru07] : parcours aléatoire ('random walk'), multiple parcours aléatoires ('multiple random walks'), méthodes hybrides (qui englobe à la fois le parcours aléatoire et la technique d'inondation) [Gka05], techniques basées sur des informations statiques [Mas05b], les indices de suivi ('forwarding indices') [Cre02], et l'introduction de la sémantique des informations [Zei05, Tal05, Cre03]. Une autre solution consiste à sauvegarder

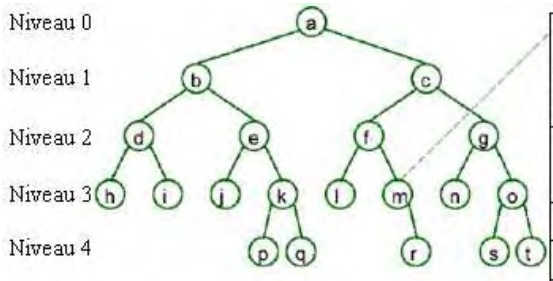
les réponses des requêtes précédentes (créer une sorte de mémoire) au niveau de chaque pair [Kal02].

Toutes ces solutions ont essayé d'une façon ou d'une autre de réduire le nombre de messages générés. Afin de comprendre comment certains auteurs arrivent à réduire le nombre de messages, nous décrivons quelques solutions. Nous commençons, par exemple, par la solution proposée par [Kal02] qui utilise une mémoire de stockage. En effet, dans cette dernière, chaque pair garde une trace des voisins qui ont répondu à une requête donnée afin de réutiliser ces informations lors des prochaines requêtes. Le pair détenant une nouvelle requête R_n va la comparer aux anciennes R_i qu'il a déjà mémorisées. Si le pair trouve une similarité entre R_n et une R_1 de R_i alors il envoie R_n vers les voisins qu'il a dans sa mémoire et qui ont déjà contribué à une réponse pour R_1.

D'autres solutions proposent la réduction de la valeur du TTL; soit en choisissant sa valeur selon une certaine probabilité soit en la fixant à une constante maximale par pair (fixer la valeur à 1 par exemple, par la suite le nombre de messages générés sera au grand maximum de k*TTL). Toutefois, ce type de réductions excessives empêche dans la plupart des cas d'obtenir une bonne réponse au vu du risque de ne pas parcourir la majorité des pairs dont parmi eux plusieurs peuvent avoir bien évidemment la réponse ou une partie de la réponse. Le travail [Yan03] améliore les performances du routage avec une méthode itérative. Le routage est répété de façon aléatoire (ou aveugle) autant de fois que le nombre fixé par le TTL jusqu'à obtenir des résultats satisfaisants. Ces solutions aident à contourner les limitations des systèmes pair-à-pair non-structurés mais jusqu'au jour d'aujourd'hui, elles ne permettent pas une réduction satisfaisante du nombre de messages et elles augmentent le risque des fausses réponses. Ces inconvénients ont incité les chercheurs à penser à d'autres solutions en s'appuyant sur les systèmes pair-à-pair structurés et les systèmes hybrides.

2.2.2. Approches basées sur les systèmes pair-à-pair structurés

Les systèmes pair-à-pair structurés [Ali07, And02, Che05, Spe03, Cai03, Mis04, Opp03, Tal07b, Xia05] sont basés sur un mécanisme d'indexation distribué permettant de trouver efficacement une source de donnée recherchée. Les pairs sont organisés selon une structure virtuelle qui peut être sous différents formats [Alk10]. En effet, cette structure peut être présentée par (i) un arbre [Jag05], (ii) un hypercube [Sch02], (iii) un espace cartésien de *d* dimensions [Rat01] ou (iv) un anneau [Sto01]. Ces topologies influent sur la méthode que chacune adopte pour gérer ces propres données ainsi que pour le routage d'une requête de découverte de sources de données. Dans les figures : 1, 2, 3 et 4, nous donnons quelques exemples illustratifs de ces quatre formes de systèmes.



Architecture d'index d'un arbre binaire

Nœud m: niveau = 3, nombre = 6, parent = f, fils gauche = nul, fils droite = r, l'adjacent gauche = f, l'adjacent droit = r

Table de routage gauche:

	Nœud	Fils gauche	Fils droit	Borne inférieure	Borne supérieure
0	l	nul	nul	$l_{Inférieure}$	$l_{Supérieure}$
1	k	p	q	$k_{Inférieure}$	$k_{Supérieure}$
2	i	nul	nul	$i_{Inférieure}$	$i_{Supérieure}$

Table de routage droite:

	Nœud	Fils gauche	Fils droit	Borne inférieure	Borne supérieure
0	n	nul	nul	$n_{Inférieure}$	$n_{Supérieure}$
1	o	s	t	$o_{Inférieure}$	$o_{Supérieure}$

Figure 1 : Exemple de système pair-à-pair structuré avec une topologie en arbre : exemple de Baton [Jag05]

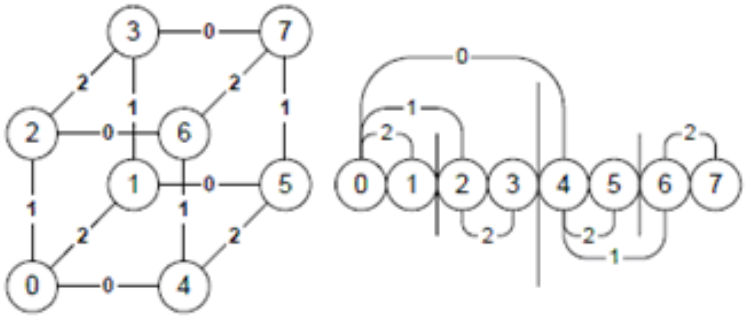


Figure 2 : Exemple de système pair-à-pair structuré avec une topologie en hypercube [Sch02]

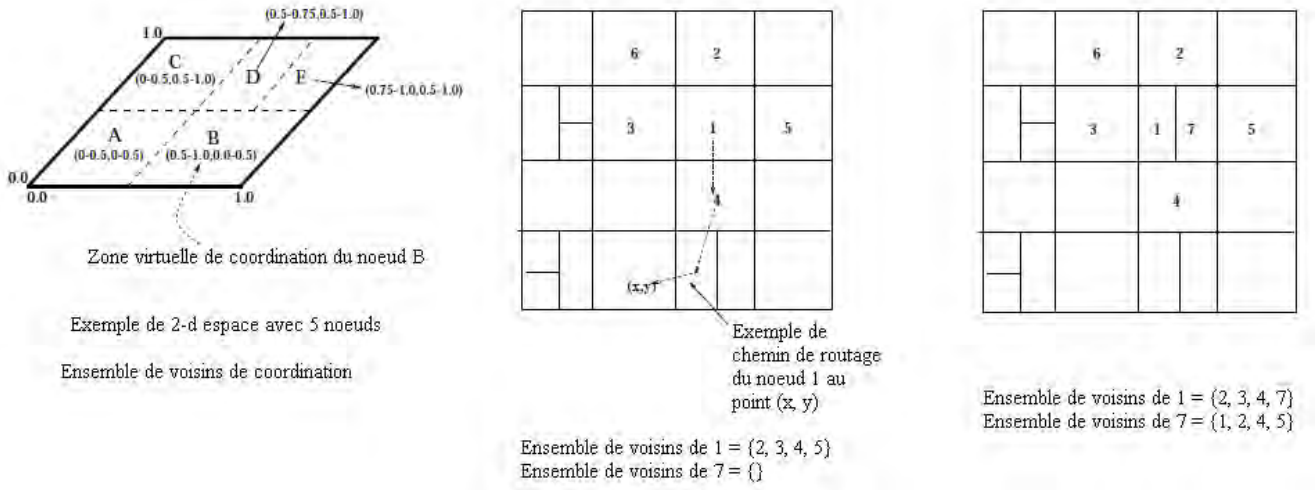


Figure 3 : Exemple de système pair-à-pair structuré avec une topologie en espace cartésien de *d* dimensions : exemple de CAN [Rat01]

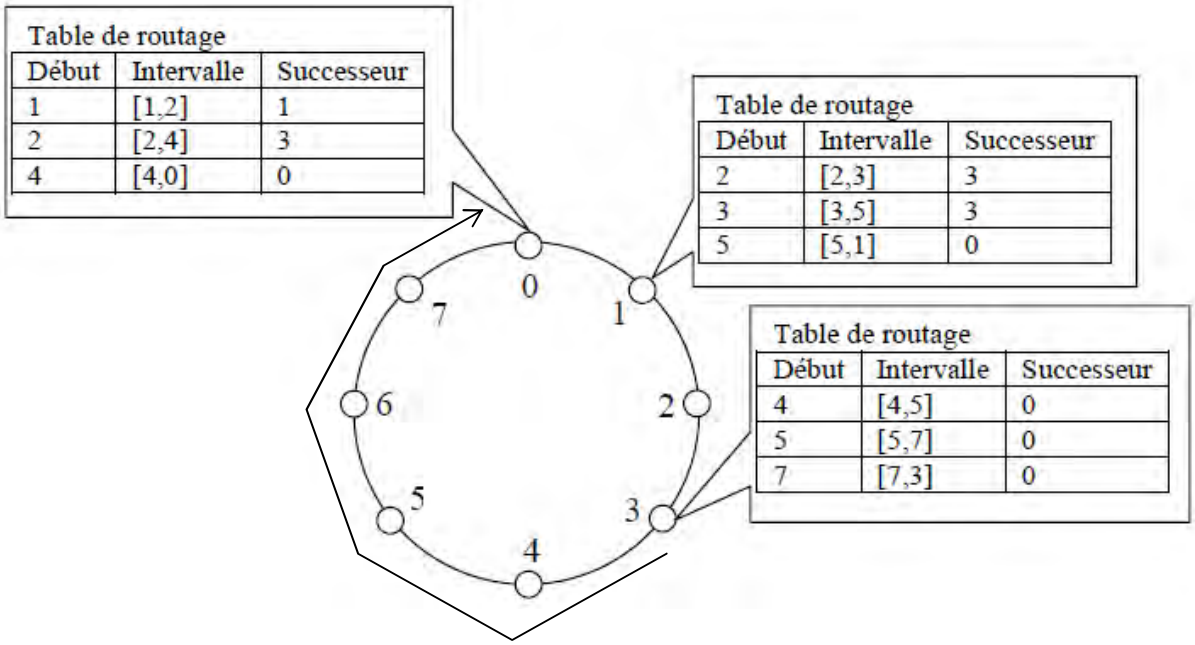


Figure 4 : Exemple de système pair-à-pair structuré avec une topologie en anneau : exemple de Chord [Sto01]

Dans ce qui suit, nous abordons, tout d'abord, la structure virtuelle de chaque type de système cité. Ensuite, nous décrivons brièvement le mécanisme de recherche de chacun ainsi que le principe de mise à jour de tels systèmes. Enfin nous présentons une comparaison synthétique des différents systèmes.

Dans la structure virtuelle en arbre, nous citons l'exemple de Baton [Jag05]. Chaque pair possède un identifiant logique lié à son niveau et son ordre dans ce niveau. Il possède également un identifiant physique (son adresse IP). Il garde un lien avec ses parents ainsi qu'avec ses voisins par cet identifiant physique. Dans un tel système, chaque pair est responsable d'un ensemble de valeurs. Par exemple, dans la figure 1 le pair *k* est responsable de l'intervalle de valeurs [$k_{Inférieure}$, $k_{Supérieure}$]. Ce même pair représente le parent de *p* et *q* et donc son intervalle de valeurs doit être compris entre leurs deux intervalles de valeurs, i.e. son intervalle ([$k_{Inférieure}$, $k_{Supérieure}$]) doit être égal à [$p_{Supérieure}$, $q_{Inférieure}$]. La table de routage se compose de deux parties : la partie des voisins droits et la partie des voisins gauches. Le nombre d'entrée dans la table de routage est de O(logN) avec *N* présente le nombre de pairs dans le

réseau. Les liens entre les pairs sont conçus de la façon suivante : la $n^{ième}$ entrée dans la table de routage est relié au pair $k+2^{n-1}$ du même niveau s'il s'agit de la table de routage droite et $k-2^{n-1}$ s'il s'agit de la table de routage gauche. Le mécanisme de routage d'une requête se fait de la manière suivante : si la borne supérieure (inférieure) de l'intervalle de la requête est plus grande (plus petite) que celle des intervalles des voisins alors cette requête sera envoyée vers le voisin le plus loin de façon horizontale. Par la suite, un envoi vertical de la requête sera procédé vers le voisin ou le fils (droit ou gauche). Ce processus de routage s'exécute de façon récursive jusqu'à trouver le bon résultat de la requête avec une complexité de O(LogN).

Concernant la structure virtuelle en hypercube illustré dans la Figure 2, les pairs sont organisés dans un hypercube. Chaque dimension possède *b* pairs (la base de l'hypercube). Nous prenons par exemple, la valeur de *b* à 2. N est le nombre de pairs dans la structure. Δ Est le diamètre[2] de l'hypercube $\Delta = O(\log_b N)$. Dans l'exemple de la Figure 2, $\Delta = \log_2 8 = 3$, ce qui est interprété par le plus long chemin est trois. Le nombre de dimensions est d+1. Chaque pair possède (b-1) (d+1) voisins ordonnés symétriquement grâce à des liens reliant tous ces pairs. Ces liens sont sous forme de numéros allant de 1 à *b*. Dans l'exemple cité, il y a un lien labélisé 2 entre les pairs 6 et 7 ce qui veut dire que le pair 6 et le $2^{ème}$ voisin du pair 7 et vice versa.

Le routage de requêtes de découverte de sources de données dans un hypercube [Sch02] se fait en envoyant cette requête vers les voisins tout en fixant un TTL. Si un pair reçoit une requête d'un voisin à lui noté i, alors il doit envoyer cette requête à des voisins ayant des numéros de liens plus petits que celui avec le pair i. La complexité de recherche en employant cette méthode est de l'ordre de $O(\log_b N)$ sauts. À titre d'exemple, en traitant toujours le même exemple cité par la figure 2, si le pair 7 envoie la requête à ses trois voisins les pairs 3, 6 et 5 alors le pair 3 peut à son tour envoyer la requête à chacun de ses voisins 2 et 1 vu que ses liens avec ces deux derniers sont labellisés

respectivement 2 et 1 tout en étant supérieurs au 0 son lien avec le pair initiateur[3] 7. Par contre, le pair 6 ne peut rien envoyer puisque ses liens avec ses voisins sont labellisés 0 et 1 bien inférieur à 2 (son lien avec le pair initiateur 7). Et par la suite, le pair 5 ne peut envoyer la requête qu'au pair 4.

Décrivons maintenant la structure virtuelle espace cartésien de *d* dimensions correspondant à la figure 3 représentant l'exemple de CAN (Content Adressable Network) [Rat01]. Le principe consiste, tout d'abord, à découper un espace de d dimensions en un ensemble de zones séparées les unes aux autres. Chaque pair est responsable d'un ensemble de stockage, il utilise sa propre zone pour stocker tous les couples (Clés, Identifiant). Cet espace de stockage est un espace cartésien de d dimensions indépendant du réseau physique. Chaque clé est un point de coordonnées c = (c1, c2, …, cd). La distribution des clés via les zones se fait selon une fonction de hachage.

Concernant le routage, CAN utilise l'algorithme de type glouton 'Greedy Algorithm'. Dans ce type de routage (système), chaque pair stocke les informations relatives à *2d* voisins. Le nombre moyen de sauts pour découvrir un pair responsable d'une source de données est de $O(dN^{1/d})$ avec *N* nombre de pairs dans le système. L'exemple de la Figure 3 montre une illustration de ce processus de routage. Nous supposons un espace cartésien de 2 dimensions. Si un pair, 1 par exemple, souhaite chercher une source de données présentée par une clé (x, y) alors il envoie la requête de découverte vers le pair voisin ayant la zone la plus proche du pair responsable de la zone de cette clé (e.g., le pair 4 dans la figure 3). Puis, le pair 4 envoie la requête, à son tour, vers le pair voisin le plus proche de la zone de la clé (x, y) et ensuite le processus se réitère de cette façon jusqu'à trouver le pair responsable à cette clé.

Finalement, le dernier type de structure virtuelle que nous abordons est la structure en anneau [Sto01]. Les systèmes pair-à-pair se basant sur une

---

[2] Plus longue distance entre deux pairs.

structure virtuelle sous forme d'anneau sont les plus populaires des systèmes pair-à-pair. Nous choisissons dans la figure 4 l'exemple de Chord [Sto01] que nous décrivons tout au long de cette section. Chord est un protocole de routage qui permet de localiser des clés pointant sur des pairs qui stockent des données. Dans Chord, tous les pairs sont organisés selon un anneau virtuel. La localisation est effectuée en associant une clé à chaque valeur. Une valeur peut être sous forme d'adresse ou document ou une entité de données, etc. Ainsi, un couple de (Clé, Valeur) est stocké sur le pair responsable de cette clé.

Le processus de routage dans Chord [Sto01] se fait selon la méthodologie suivante. Une fonction de hachage affecte à chaque pair et clé un m-bit identifiant en employant une base de hachage [Fip95]. Cet identifiant est choisi pour un pair en se servant de son adresse IP. La fonction de hachage attribue les clés aux pairs d'une façon particulière que nous décrivons brièvement dans ce qui suit. Les identifiants sont organisées selon un cercle modulo 2^m et une première clé k est associée au premier pair qui possède un identifiant égale à k ou qui suit l'identifiant de k. Dans ce cas, ce pair est appelé le pair successeur de la clé k (successeur(k)). Dans l'exemple de la figure 4 présente un cercle d'identifiant dont m = 3. L'anneau virtuel possède trois pairs : 0, 1 et 3. Le successeur de l'identifiant 3 est 0, donc la clé 3 doit être localisée au niveau du pair 0 et de même pour les autres. Chord se distingue par rapport aux autres protocoles pair-à-pair de recherche par un routage simple de l'ordre de O(logN), avec *N* le nombre de pairs dans le réseau. Chord permet aussi de gérer le processus de réplication de données souvent utilisé dans les systèmes dynamiques en stockant une donnée répliquée dans deux clés différentes. Toutefois, cette performance perd son importance quand l'information devient obsolète.

Dans un environnement à grande échelle une méthode de découverte de sources de données doit être efficace et scalable. De plus, le mécanisme de la

---

[3] Nœud qui lance la propagation d'une requête de découverte de sources de données vers les autres nœuds du système.

maintenance de la structure virtuelle dû à la dynamicité des pairs doit également être efficace. En effet, il ne faut pas qu'une méthode passe principalement son temps à maintenir sa structure. Maintenir le système revient à gérer l'arrivée et le départ des pairs avec un minimum de perturbations possible. Lors d'une connexion (un pair joint le système) certaines clés affectées à son successeur doivent être affectées au pair entrant. Dans le cas contraire, lors d'une déconnexion si un pair quitte le système toutes ses clés sont affectées à son successeur. La maintenance du système selon Chord nécessite $O(\log^2 N)$ messages après la connexion / déconnexion d'un pair du système.

En conclusion, Chord est scalable et s'adapte de façon efficace pour le processus de découverte de sources de données [Sto01]. Cependant, le coût de communication et de maintenance est prohibitif lorsque le nombre de pairs est important et quand il existe un processus continu de connexion et de déconnexion ce qui est le cas à grande échelle.

Discussion

En comparant ces quatre systèmes, nous constatons que les tables de hachage distribuées (THD) utilisées dans Chord mais aussi dans Pastry [Row01] ont l'avantage de redistribuer les informations des placements des données et pas les données elles mêmes avec leurs valeurs (e.g., algorithme de Baton). Plus précisément, une THD permet le stockage et l'accès aux pairs par une fonction qui associe chaque valeur à une clé (clé, valeur). Ainsi, la recherche se fait en appliquant une fonction dite de 'hachage' à des clés. Toutefois, du point de vue de la maintenance et le nombre de messages nécessaires pour maintenir le système, les THD tel que Chord et CAN sont moins efficaces que l'arbre de Baton et Hypercube. Les pairs sont organisés selon la même fonction de hachage avec des clés qui facilitent la localisation des données (ou plutôt des pairs responsables à ces données). La complexité de routage dans une THD est souvent de l'ordre de $O(\log N)$ avec N la dimension de l'espace des clés. Dans Chord par exemple, comme le premier système par-

à-pair structuré (Figure 4), les pairs sont organisés en forme d'un anneau dimensionné suivant leurs clés. Chaque pair stocke l'index des clés. Un pair qui vient de joindre le réseau prend la responsabilité de quelques clés qui étaient affectées auparavant à un de ses voisins. En contre partie, un pair qui quitte le système donne la responsabilité des ses clés à ses pairs voisins affectés par son départ. Toutefois, CAN (Figue 2) essaie de réduire le nombre de voisins d'un pair à une constante. Les pairs dans CAN sont organisés selon un espace en d-dimension. Autrement dit, chaque clé de pair est constituée de *d* nombres et présente un point dans l'espace *d* dimensionnel. Dans chaque dimension, chaque pair est connecté à son voisin qui le précède ainsi que celui qui le suit dans chaque dimension et de cette façon là il aura O(d) voisins. L'espace *d* dimensionnel est réparti de façon équivalente entre les pairs du système. Chaque pair est responsable de toutes les données des clés des points de son propre sous-espace. La complexité de recherche ('look-up') est $O(N^{1/d})$ en temps et nombre de messages nécessaires aussi. La dynamicité des pairs dans CAN n'affecte pas trop le système vu que l'arrivée ou le départ d'un pair n'affecte que O(d) pairs dans le système, ce qui justifie l'utilité de réduire le nombre de voisins à une constante. Un nouveau pair ajouté au système doit contacter un des pairs existants. Ce dernier divise son sous-espace en deux parties. Il garde la responsabilité d'une des deux parties et affecte l'autre au nouveau pair. CAN a eu des extensions en employant plus qu'une fonction de hachage dans le but de pouvoir maintenir la réplication et par la suite réduire le coût de la recherche et assurer la tolérance aux pannes [Tru07]. Chord améliore le pouvoir de passer à l'échelle (scalabilité) au niveau du processus de découverte en réduisant les informations que possède chaque pair sur tous les autres pairs. En effet, un pair ne doit savoir que quelques informations de routage sur les autres pairs, qui présente une information distribuée. La fonction de hachage s'exécute en mettant en communication un nombre réduit de pairs. En effet, dans un réseau de N-pairs, chaque pair maintient l'information seulement de O(logN) d'autres pairs, et une recherche nécessite O(logN) de messages. En conséquence, l'arrivée / départ d'un pair engendre $\log^2 N$ messages pour la mise à jour du système.

2.2.3. Approches basées sur les systèmes hybrides

Dans les sections précédentes, nous avons présenté les systèmes pair-à-pair non-structurés et les systèmes pair-à-pair structurés. Nous avons discuté les avantages de chacun de ces systèmes en précisant également les inconvénients. Notamment, nous avons mis l'accent sur le risque élevé de la fausse réponse et la saturation de réseau pour les systèmes non-structurés et les coûts de maintenance prohibitifs pour les systèmes structurés. Les systèmes hybrides [Yan03, Fil04, Mas05a, Pup05] essaient de combler ces limites tout en conservant les points forts de ces deux types de systèmes [May02, Koo06] notamment la recherche guidée. En d'autres termes, ces systèmes dits 'hybrides' ont pour objectifs d'avoir les bons résultats des systèmes structurés ainsi que l'autonomie et la distribution des pairs des systèmes non-structurés.

Ces systèmes sont appelés 'hybrides' vu qu'on retrouve deux types d'architectures : (i) l'architecture client / serveur entre les super-pairs et leurs pairs et (ii) l'architecture pair-à-pair généralement non-structurés entre les super-pairs. Plus précisément, les pairs sont divisés en deux catégories : les super-pairs et les pairs. Chaque pair super-pair joue le rôle d'un 'proxy' (serveur) et donc il est responsable d'un groupe de pairs (clients). Il est responsable de toutes les tâches liées à ses pairs (feuilles); la maintenance, le contrôle. Il est responsable également d'indexer leurs données, du processus de découverte de données, de récupérer les réponses, de les contrôler, etc. Il est aussi responsable de toutes les tâches liées à ses pairs (feuilles); la maintenance, le contrôle… Ces super-pairs sont souvent des pairs choisis suivant leurs caractéristiques (puissance de calcul, bande passante, etc.) afin de pouvoir achever des opérations plus ou moins compliquées citées précédemment. Deux types de serveurs super-pairs peuvent apparaître selon les besoins du système. Il y a des serveurs temporaires, c'est-à-dire les pairs du réseau peuvent devenir à tout moment des super-pairs et vice versa dépendamment des besoins du système, e.g., les systèmes FastTrack comme KaZaA. Il y a également les serveurs fixes (ou permanent) qui ne changent pas

de rôle et donc moins souple que ceux temporaires, e.g., le système eDonkey2000. Le principe de découverte dans les systèmes hybrides se fait essentiellement par un traitement local de la requête au niveau de la composante[4] recevant la requête utilisateur (la racine représentant le super-pair et ses feuilles) puis entre les super-pairs. Précisons que la recherche dans de tels systèmes est aussi basée sur l'utilisation d'un TTL. Si le pair qui vient de recevoir cette requête est un super-pair alors il va chercher la réponse au niveau de ses données indexées (comprenant toutes les données de la composante), en premier lieu. En deuxième lieu, s'il ne trouve pas la réponse nécessaire, il va s'adresser aux super-pairs voisins. Dans l'autre cas de figure, si le pair recevant la requête utilisateur est une feuille alors il va chercher la réponse localement, tout d'abord, au niveau de ses sources de données. S'il ne trouve pas la réponse alors il envoie la requête au pair super-pair et c'est à celui-ci de chercher la réponse localement au niveau de ses sources de données sinon il passe la requête à ses voisins (d'autres super-pairs). Ce processus se répète autant de fois que le TTL n'a pas encore atteint zéro. La réponse finale revient, bien évidemment, au pair initiateur. Ce type de système a l'avantage des deux types d'approches centralisée et distribuée. Il bénéficie de l'approche centralisée au sein d'une composante super-pair (architecture client / serveur) offrant entre autres une maintenance assez facile vu qu'elle se limite uniquement au nombre des super-pairs comme points d'accès. Les systèmes hybrides (super-pairs) héritent, également, de l'avantage des systèmes pair-à-pair en permettant une recherche distribuée. Cependant, le processus de découverte de sources de données dépend du type du système pair-à-pair employé (topologie entre les super-pairs) ainsi que la méthode de découverte de sources de données adoptée. D'un autre coté, les systèmes hybrides risquent aussi la saturation des pairs super-pairs (e.g., goulets d'étranglement) comme tous les systèmes adoptant la centralisation. Ils sont également moins tolérants aux pannes à travers le point unique de défaillance ('single point of failure'). Ceci intervient dans le cas où

[4] Une composante est constituée d'un nœud super-pair et de ses nœuds feuilles.

un super-pair se déconnecte du système. Pour remédier à ces deux derniers problèmes, certains ont pensé à la duplication d'un super-pair en *k* super-pairs [Yan03]. Toutefois, ceci augmente le coût de communication après la connexion / déconnexion d'un pair vu que tous les k super-pair doivent être informés par la mise à jour des pairs. En conséquence, ceci augmente le nombre de messages nécessaires pour cette mise à jour.

2.3. Comparaison

Dans cette section, nous définissons les critères de comparaison afin de comparer entre les différentes méthodes existantes de découverte de ressources.

2.3.1. Critères de comparaison

Comparer les méthodes de découverte de sources de données basées sur les systèmes pair-à-pair nécessite l'étude et l'analyse de plusieurs critères au sein d'un environnement de grille. Plusieurs travaux ont été menés dans ce contexte [Mes08 et Tru07]. Ces travaux comme beaucoup d'autres citent de nombreux critères qui peuvent être étudiés dans cette perspective. Toutefois, nous nous contentons de ceux que nous pensons les plus pertinents. Ces derniers sont tout d'abord, liés aux caractéristiques majeures de l'environnement de grille, à savoir : *l'instabilité* (dynamicité des pairs du réseau) et la capacité de pouvoir *passer à l'échelle*. Ensuite, il y a également, les critères liés au processus de découverte de sources de données (indépendamment de l'environnement), à savoir : *le routage de requête*[5], *la complexité* de recherche, *la maintenance* du système et *la fiabilité* du système. Nous définissons ces critères dans ce qui suit.

Critères liés aux caractéristiques majeurs de la grille :

- *L'instabilité* : au sein d'un environnement de grille, l'instabilité des pairs est une contrainte que nous ne pouvons pas contourner. En effet,

[5] Deux types de requête : (i) la requête de l'utilisateur qui est la requête entière fournie par l'utilisateur que nous l'appelons le long de ce document requête utilisateur (R) et (ii) la requête qui présente une partie de la requête utilisateur qui peut être appelé aussi message que nous l'appelons le long le ce document requête.

chaque pair peut rejoindre ou quitter le système à tout moment. Nous noterons à chaque fois la capacité de chaque méthode à prendre en compte l'instabilité du système tout en gardant sa performance.

- *Le passage à l'échelle* : ce critère indique si la méthode de découverte de sources de données est capable de maintenir le même rendement tout en augmentant le nombre le nombre de pairs et d'utilisateurs du système sans détérioration des performances (e.g., saturation au niveau d'un pair provoquant un goulet d'étranglement).

Critères liés au processus de la découverte de sources de données :

- *Le routage de requête* : ce critère nous renseigne sur la façon dont une requête de découverte de sources de données est envoyée vers et à travers les pairs et la façon dont la réponse est retournée aussi. Le routage dépend beaucoup de la topologie du système (par-à-pair non-structuré, pair-à-pair structuré ou hybride).
- *La complexité* : ce critère renvoie la complexité de la méthode de découverte de sources de données en termes de temps de réponse ainsi que de messages générés.
- *La fiabilité* : ce critère indique si la méthode de découverte de sources de données est capable de retourner dans tous les cas une réponse.
- *La maintenance* : ce critère précise si la méthode de découverte de sources de données exige un coût de maintenance important tout en opérant dans un environnement dont les pairs quittent / rejoignent fréquemment le système.

Après avoir cité ces quelques critères, nous présentons une comparaison de ces trois types d'approches. En particulier, nous détaillons le processus de recherche par mots clés au sein de ces trois types de systèmes.

2.3.2. Discussion

L'étude des trois types de systèmes précédents a montré que le processus de routage de requêtes de découverte de sources de données dépend

essentiellement de la topologie du système pair-à-pair employé ainsi que le protocole d'adressage utilisé.

Au niveau du premier type de système; *les systèmes pair-à-pair non-structurés* [Gnu00, Abd05, Iam01, Iam04, Jea08, Tal05, Kal02, Cha03], la recherche est procédée essentiellement par le principe d'inondation. Le pair qui détient la requête (pair initiateur) envoie sa requête à tous ses voisins. Cet envoi est réalisé avec (i) un coût de communication assez élevé et (ii) un coût de traitement local de requête faible par rapport à ce premier voir négligeable et donc il n'est pas souvent étudié. En effet, l'envoi de la requête à tous les voisins nécessite un nombre important de messages proportionnellement au nombre d'utilisateurs du réseau et parfois même exponentiel relativement à la profondeur de la recherche. De plus, le mécanisme d'inondation utilisé par ce type de système risque facilement une saturation au niveau des pairs du réseau pair-à-pair (e.g., des goulets d'étranglement). Par la suite, bien que ces systèmes pair-à-pair non-structurés se distinguent par une autonomie de pairs remarquée, ce qui rend le processus de maintenance plus efficace, ils présentent un point faible par rapport au problème du passage à l'échelle. Beaucoup d'efforts ont été déployés pour combler cette lacune comme par exemple la réduction du trafic de diffusion, réduire la zone la diffusion de la requête de découverte de sources de données et ne l'envoyer qu'à une sélection bien déterminée de pairs voisins qui sont souvent réputés d'être les meilleurs en termes de capacité à retourner le bon résultat (moins de risque de fausses réponses). Du point de vue de la maintenance du système, les systèmes pair-à-pair non-structurés présentent de bonnes propriétés En effet, lors de l'ajout d'un pair, il suffit juste d'établir ses liens avec un ou plusieurs autres pairs. Par ailleurs, à la déconnexion d'un pair, ce dernier pair n'a pas à prévenir quiconque de cette déconnexion. Ceci est point positif vu l'instabilité des pairs dans les environnements à grande échelle.

Dans *les systèmes pair-à-pair structurés*, les pairs sont organisés suivant une certaine topologie. La plupart du temps, une table de hachage distribuée

(THD) est utilisée telle que Chord [Sto01]. La recherche au niveau de ces tables est faite en un nombre de messages qui croit de façon logarithmique avec le nombre de pairs. Ce type de système a pu assurer un degré de succès et de popularité assez important vu l'avantage de la décentralisation qu'il offre tout en palliant les limites des systèmes cités précédemment. En effet, contrairement aux systèmes pair-à-pair non-structurés, les systèmes pair-à-pair structurés facilitent le processus d'exécution de la requête de découverte de sources de données et le rend donc plus efficace grâce à leur structure bien déterminée. Autrement dit, la requête de découverte suit une technique de recherche guidée qui rend cette dernière plus efficace générant ainsi moins de risque de fausses réponses. Les systèmes pair-à-pair structurés ont aussi prouvé leur efficacité par rapport au passage à l'échelle (scalabilité) surtout en termes de charge de trafic. Néanmoins, afin de maintenir leur structure rigide, ils génèrent un coût de maintenance élevé ce qui constitue l'inconvénient majeur de ces systèmes. En effet, les pairs peuvent quitter ou rejoindre le système à tout moment (dynamicité des pairs dans de tels systèmes) surtout en présence de l'effet 'Churn'[6] [Tru07 et Sam09].

Vu les limites des systèmes structurés et non-structurés, les systèmes décentralisés hybride ont vu le jour. Le principe de cette approche consiste à considérer des pairs super-pairs [Sen02, Yan03, Fil04, Mas05a, Pup05, Lia06]. Ce sont des serveurs dont chacun est responsable d'un groupe de pairs clients. Cela évite le fait d'avoir un seul serveur central, ce qui évite par la suite les limites de la centralisation du système (tolérance aux pannes, fiabilité, goulets d'étranglement, un seul point de défaillance, etc.). Dans de tels systèmes, le mécanisme de communication entre les pairs clients et les pairs serveurs (entre les feuilles et les super-pairs) est souvent simple et ne peut dépendre que de la méthode de découverte de sources de données employée. Cependant, quand il s'agit du mécanisme de communication entre les super-pairs, l'efficacité du mécanisme dépend de la topologie qui relie les super-pairs entre eux ainsi que la méthode de découverte de sources de données utilisée. S'ajoute à cela, que ces systèmes sont les moins tolérants aux pannes par rapport aux systèmes pair-

[6] L'effet 'Churn' : Arrivée / départ des nœuds de façon continue.

à-pair structurés et non-structurés. En effet, si un pair super-pair quitte le système tous ses pairs feuilles le quittent également. Sans oublier le risque de fausses réponses si la topologie qui lie les super-pairs est non-structurée (ce qui est souvent le cas).

Dans le tableau 1, nous trouvons un résumé de tous les critères discutés.

Critères de Comparaison	Liés aux caractéristiques de grille		Liés au processus de découverte de sources de données					
	Passage à l'échelle du processus de maintenance	Passage à l'échelle de la méthode de recherche	Routage de requête	Complexité de recherche (en termes de temps)	Complexité de recherche (en termes de trafic, nombre de messages)	Robustesse	Fiabilité	Coût de maintenance en présence effet de 'Churn'
Systèmes P2P non-structurés	- Oui	- Non	- Diffusion	- O(logN)	- O(N^2)	- Degré élevé	- Risque élevé de fausses réponses ou réponses erronées	- Relativement faible
Systèmes P2P structurés	- Non	- Oui	- Routage guidé	- O(logN)	- O(logN)	- Degré bas	- Fiable	- Élevé
Systèmes super-pairs	- Oui pour les feuilles et pas pour les super-pairs	- Non	- Diffusion et - Routage guidé	- O(logS)	- O(S^2)	- Degré élevé (dépend de la topologie entre les super-pairs)	- Degré moyen de risque de fausses réponses ou réponses erronées (dépend de la topologie entre les super-pairs)	- Dépend de la topologie entre les super-pairs

(Avec *N* : nombre de pairs (nœuds) dans le réseau, *S* : nombre de super-pairs dans le réseau.)

Tableau 1 : Comparaison des approches de découverte de sources de données pair-à-pair

Les premiers travaux traitant la découverte de sources de données et basés sur les systèmes cités plus haut se basaient tous sur le mécanisme de recherche par mots clés. Toutefois, la recherche par mots-clés peut facilement aboutir à des résultats erronés, biaisés, moins précis ou manquant à cause des problèmes de sémantique [Cru04a, Liu05, Cas04, Ket10] présent dans tous les domaines comme par exemple la biologie (e.g., problèmes liés à la synonymie, homonymie, polysémie…). Nous prenons l'exemple du mot *Docteur* qui a pour sens un médecin dans les sources de données liés au domaine médical alors que ce même mot dans un domaine biologique désigne un employé possédant un diplôme de doctorat dans un laboratoire de biologie (biologiste). Dans ce cas, le mécanisme de recherche par mot clés peut fournir un résultat erroné. Pour ces raisons, les méthodes de recherche doivent se baser sur des techniques plus fiables, traitant plus l'aspect sémantique. Dans la section suivante, nous introduisons les méthodes prenant en compte la sémantique dans la découverte de sources de données.

3. Mécanisme de découverte de sources de données tenant compte de la sémantique

Nous présentons dans cette section une synthèse sur les approches existantes de découverte de sources de données en tenant compte des problèmes liés à l'hétérogénéité sémantique. Dans notre étude bibliographique, nous avons constaté que c'est souvent les problèmes liés à l'hétérogénéité des structures des données (e.g., différence de format, structure, complexité des schémas de données) qui ont été largement étudiés dans la littérature [Cru04b, Zhu04, Liu05, Cas04, Rey03, Sar04, Sio05].

La plupart des approches de découverte de sources de données se basent sur la recherche par mots clés, tel que dans PeerDB [Sio05]. Toutefois, ce type de recherche reste insuffisant et / ou inefficace vu les divers problèmes liés à l'hétérogénéité sémantique (synonymie, polysémie, abréviation…) [Pir08a, Pir08b, Com09] qui peuvent exister au sein de n'importe quel domaine dans un environnement à grande échelle. Ces problèmes deviennent de plus en plus

délicats dans les domaines où la réponse fournie à un utilisateur doit être d'un niveau de précision assez important, comme par exemple dans le domaine biomédical où la santé et le bien être des êtres humains est en jeu. Plusieurs travaux de recherche se sont intéressés à traiter la sémantique dans le processus de découverte de sources de données. Certains travaux ont proposé d'établir des correspondances entre les mots clés des schémas ou ontologies [Nav05, Jon08]. La maintenance de ces correspondances est une tâche lourde et fastidieuse. Beaucoup de travaux se sont basés sur un schéma global ou une ontologie centrale qui gère toutes les sources de données. Néanmoins, concevoir un schéma global ou une telle ontologie globale demeure une tâche complexe à réaliser devant le nombre gigantesque de sources de données dans n'importe quel domaine et notamment le biomédical. A partir de cette constatation est apparue l'émergence de la notion d'ontologie de domaine [Nav05]. Une ontologie est associée à chaque domaine et des relations de 'mappings' définissent les liens de correspondances entre ces dernières. Elles permettent notamment de définir des relations d'équivalence entre les concepts appartenant à des ontologies différentes. Dans la suite nous présentons : (i) les approches basées sur des correspondances entre mots clés (e.g., PeerDB [Sio05], 'WSPD' [Ban04] et [Li05]) et (ii) les approches utilisant une ontologie globale (e.g., PEPSINT [Cru04b] et APPA [Akb04] et Bibster [Haa04]) ou plusieurs ontologies (e.g., [Alk08], [Hei04] et [Zhu04a]). Nous explicitons d'avantages cette classification [Ket11a] dans la suite.

3.1. Approches basées sur des correspondances entre mots clés

Les solutions classiques de découverte de ressources étaient basées, comme nous l'avons signalé plus haut, sur la recherche par mot clés. Dans **PeerDB [Sio05]**, les métadonnées utilisées par le processus de découverte de sources de données sont présentées sous une forme relationnelle permettant le partage de ces métadonnées. Le routage se base sur des correspondances entre les mots clés. Vu les problèmes de l'hétérogénéité sémantique des données [Pir08a, Pir08b et Com09], ce type de recherche risque de retourner de faux

résultats ou des réponses incomplètes. S'ajoute à cela, la limite du système au niveau des données supportées qui ne peuvent être que relationnelles. En plus, PeerDB repose sur l'approche non-structurée des systèmes pair-à-pair, ce qui abouti à un processus de localisation non fiable (résultat existant mais non retourné).

Le service de découverte de sources de données dans le système '**WSPD**' **[Ban04]** introduit la sémantique dans la découverte de services web. Il utilise également des correspondances entre mots clés. Il introduit la sémantique en employant le QDN (Querical Data Network) où l'identité d'un pair est identifiée par ses données. Néanmoins, pour diffuser les requêtes de découverte de sources de données, le service emploie un mécanisme de diffusion par inondation. [Ban04] se base sur une architecture pair-à-pair non-structurée basée sur le protocole Gnutella [Gnu00]. D'autres travaux tel que **[Li05]** utilisent plutôt une architecture hiérarchique pour permettre une découverte de ressource tout en permettant le passage à l'échelle.

La définition des correspondances entre mots clés n'est pas une tâche aisée et ces liens de correspondance sont nombreux. De plus, la maintenance de ces liens est une tâche ardue dédiée à l'administrateur du système. Les approches qui adoptent le mécanisme de recherche en se basant sur des correspondances entre mot clés traitent généralement les problèmes liés à la structure des données et ne prennent pas en considération les problèmes liés à l'hétérogénéité sémantique [Pir08a, Pir08b et Com09] sous tous ses aspects (elles traitent la sémantique de façon partielle). Ceci augmente le risque de retourner des résultats erronés ou incomplets. Ainsi, d'autres solutions ont été proposées comme le recours à des ontologies [Nav05 et Jon08]. Nous détaillons l'utilisation de la notion d'ontologie dans ce qui suit à travers quelques systèmes illustratifs.

3.2. Approches utilisant une ou plusieurs ontologies

3.2.1. Approche utilisant une Ontologie globale

Plusieurs systèmes traitent le problème de l'hétérogénéité sémantique [Pir08a, Pir08b, Com09] en utilisant soit un schéma global soit une ontologie globale utilisée comme un schéma global pivot. Par exemple, dans APPA [Akb04], les pairs voulant s'échanger des données entre eux se partagent un schéma global appelé CSD (Common Schema Description). Ce type de schéma n'est pas évident à concevoir dans un tel type d'environnement pair-à-pair distribué à grande échelle vu qu'il est partagé par un ensemble de pairs hétérogènes [Alk08]. **PEPSINT [Cru04b**] traite la sémantique par l'introduction d'une ontologie centrale qui contrôle la totalité des pairs et les correspondances sémantiques qui décrivent toute relation de sémantique entre les éléments des schémas des pairs. Ce système repose sur une topologie pair-à-pair hybride. Comme nous avons pu le constater dans la section décrivant les systèmes pair-à-pair hybrides, l'utilisation de tels systèmes génère une forte probabilité de fausses réponses en plus du risque de non fiabilité en cas de panne d'un super-pair. Tout comme PEPSINT, Bibster **[Haa04]** utilise une ontologie globale commune entre tous les pairs. Le système de routage se base sur la similarité sémantique entre la requête et la description d'expertise de chaque pair pour pouvoir sélectionner les pairs pertinents.

Toutefois, concevoir un schéma global ou une ontologie globale que tous les pairs du réseau peuvent partager n'est pas envisageable et reste une tâche difficile à réaliser jusqu'au jour d'aujourd'hui. En effet, le nombre de sources de données est gigantesque dans n'importe quel domaine et leurs mises à jour quotidienne est un véritable problème. Afin de remédier à ces problèmes, d'autres chercheurs ont envisagé d'autres solutions telles qu'adopter la notion d'ontologie de domaine où une ontologie est associée à chaque domaine [Nav05]. Des liens de correspondance dits 'mappings' entre ces ontologies sont alors définis.

3.2.2. Approche utilisant plusieurs ontologies de domaines ou schémas

Etant donné que l'utilisation d'une ontologie globale ne constitue pas une solution satisfaisante, plusieurs chercheurs ont proposé l'utilisation de plusieurs ontologies de domaines pour la prise en compte des problèmes lié à l'hétérogénéité sémantique. Dans cette solution, une ontologie est associée à chaque domaine [Nav05]. Un schéma local propre à chaque pair peut également décrire la source de données. Pour que les ontologies de domaines ou les schémas locaux des pairs puissent se partager les données entre eux et communiquer, des relations de correspondances sémantiques, appelé souvent dans la littérature 'mapping', doivent être établies entre ces ontologies de domaines.

[Alk08] Traite le problème de l'hétérogénéité sémantique lors du processus d'une découverte efficace de sources de données dans un système pair-à-pair (protocole Chord) en présentant les données de façon plus explicite et en dupliquant les ontologies de domaines au niveau de chaque pair. Les ontologies sont considérées comme les seules interfaces de communication (interaction) pouvant exister entre les pairs. Néanmoins, dupliquer les ontologies de domaines au niveau de chaque pair est une tâche non envisageable dans un environnement distribué à grande échelle comme la grille mettant en jeu un nombre gigantesque de pairs et de domaines voire ontologies de domaines. L'autre problème réside dans le fait que les domaines peuvent évoluer. Ainsi, la mise à jour des ontologies, associées aux domaines, pose problème dans l'approche proposée dans [Alk08]. De plus, les relations de correspondances sémantiques entre les ontologies de domaines n'ont pas été traitées dans [Alk08] ce qui complique le processus de découverte entre les ontologies.

[Hei04] Traite également la découverte de sources de données sémantique dans un environnement de grille. Dans cette solution, chaque pair du réseau dispose des métadonnées de ces sources. Il possède également sa propre

ontologie présentée sous forme de graphe orienté appelé DAG (Directed Acyclic Graph). Puis, la distribution de graphes locaux DAGs est faite à travers les pairs du réseau pair-à-pair à travers une table de hachage distribuée (THD). Ainsi, le processus de découverte consiste à la recherche du bon graphe dans cette THD.

[Zhu04a] Aussi utilise également une table THD pour la diffusion de messages entre les communautés (organisations). Autrement dit, les correspondances entre communautés sont établies via une THD. Puis, dans chaque communauté, la découverte de service (intra-communauté) est basée sur la méthode GSBSL (Greedy Search Based Service Location). Vu que les communautés sont organisées suivant une structure de THD, elles sont obligées de respecter la structure rigide de cette THD. Malgré que cette approche ait montré une efficacité au niveau de la méthode de recherche, la maintenance du système au niveau de cette approche est encore contrôlable.

XPeer [Sar04] se base sur la sémantique des données pour grouper les pairs en communautés. Ceci est fait selon les degrés de similarité entre les schémas de ces pairs. La découverte se fait sans le recours à un schéma global mais en partageant des métadonnées décrivant différents domaines sous format XML. Elle s'appuie également plutôt sur des relations de correspondances entre les schémas. Toutefois, mettre à jour ces relations est un processus complexe. Ce système se base également sur l'architecture hybride et plus précisément sur une architecture super-pair. Une telle architecture risque de générer des goulets d'étranglement. Une telle solution souffre également du manque de fiabilité au cas où le super-pair tombe en panne ou quitte le système.

Dans **Piazza [Hal03]** les 'mappings' sont établis entre les pairs par paire (deux à deux) ou par groupe. Ainsi la topologie du graphe i.e. formée par les ontologies et les 'mappings' est en chaîne. Une requête de découverte de sources de données sera par la suite guidée et suit un chemin bien déterminé

fourni par la chaîne. Cependant, en imposant une topologie rigide, il oblige de créer et maintenir ces ‘mappings’ à jour ce qui reste toujours une vraie lacune pour ce système.

Le travail de **[Fay07]** présente un système pair-à-pair de gestion de données distribuées appelé **SenPeer**. Il prend en compte la sémantique en organisant les pairs par domaine sémantique. Chaque pair peut stocker ses données sous forme d’un formalisme en structure de graphe enrichi sémantiquement avec des mots clés. En revanche, SenPeer repose sur la typologie de super-pair ce qui génère les mêmes problèmes cités plus haut.

Nous citerons également d’autres travaux tels qu’**Hyperion [Are03, Gia05].** La recherche dans un tel système ne se base pas sur un schéma global. En effet, les relations de ‘mappings’ sont illustrées dans des tables [Kem03] et des expressions décrivent toute correspondance sémantique pouvant exister entre les schémas (plutôt les éléments des schémas) des pairs. Ainsi, chaque requête écrite sur le schéma d’un pair peut être réécrite sur les schémas des pairs reliés à ce premier. Néanmoins, l’inconvénient de ce système c’est qu’il se limite à supporter que les données relationnelles. En plus, créer et maintenir les correspondances que nous venons de citer se fait manuellement exigeant l’intervention des experts du domaine à chaque fois.

3.2.3. Discussion

Tous les différents systèmes que nous venons de présenter essayent d’une manière ou d’une autre de prendre en compte la sémantique au niveau du processus de découverte de sources de données. Les premières solutions proposent d’établir des correspondances entre les mots clés pour traiter la sémantique. La définition de ces correspondances puis leur mise-à-jour est une tâche complexe que doit réaliser un administrateur. De plus, ces correspondances sont généralement pauvres en sémantique. Pour avoir une sémantique plus riche, d’autres solutions proposent l’utilisation d’un schéma global ou d’une ontologie globale comme schéma pivot. Toutefois, la

construction d'une telle ontologie est utopiste dans un environnement à grande échelle où des sources de données sont ajoutées quotidiennement. Ainsi, plusieurs auteurs proposent d'utiliser des ontologies de domaine. Chaque ontologie est associée à un domaine précis. Ensuite, des relations de 'mapping' sont définies afin de traduire un concept d'une ontologie Oi vers un autre concept d'une ontologie Oj avec i ≠ j. Toutes les approches utilisant plusieurs ontologies imposent une topologie particulière entre les 'mapping' d'ontologie. Par exemple, la topologie centralisée PEPSINT [Cru04b], Bibster [Haa04], APPA [Akb04], la topologie deux-à-deux dans Piazza [Hal03], une topologie tous vers tous dans PeerDB [Sio05], XPeer[Sar04], SenPeer [Fay06], table de 'mapping' dans Hyperion [Are03, Gia05], etc. Nous savons que la définition des relations de 'mapping' est une tâche difficile et fastidieuse. Ainsi, il n'est pas envisageable de laisser cette tâche à un administrateur du système alors qu'il existe des ontologies et des 'mappings' disponibles sur Internet dont la topologie est quelconque.

	Type de système pair-à-pair sous-jacent	**Nombre d'ontologies ou de schémas pivots**	**Topologie des 'mappings' entre ontologies / schémas**
PeerDB	Hybride	Plusieurs	Tous vers tous (Il existe un 'mapping' entre chaque schéma de sources de données)
PEPSINT	Hybride	1	Sans objet
Bibster	_[7]	1	Sans objet
APPA	Adapté à tout type de système	1	Sans objet

[7] _ : Critère non traité selon notre connaissance.

	pair-à-pair		
Alk08	Structuré	Plusieurs	Tous vers tous
XPeer	Hybride	Plusieurs	Tous vers tous
Piazza	Hybride	Plusieurs	Chaîne (les pairs sont connectés deux à deux)
SenPeer	Hybride	Plusieurs	Tous vers tous
Hyperion	_	Plusieurs	Les ‘mapping’ entre les sources de données sont représentés dans une table de correspondance
GSBSL	Structuré (THD)	Plusieurs	Tous vers tous
[Hei04]	Structuré(THD)	Plusieurs	Tous vers tous
WSPD	Non-structuré	Plusieurs	Tous vers tous

Tableau 2 : Comparaison des différents systèmes de découverte de source de données traitant l’aspect sémantique

4. Conclusion

Dans ce chapitre nous avons présenté un état de l’art sur les méthodes de découverte de sources de données. Ces méthodes ont débuté avec un mécanisme de recherche par mots clés (i.e., sans prendre en compte l’aspect sémantique). Ces méthodes étaient basées, initialement, sur des approches centralisées. Ces méthodes risquent un goulet d’étranglements au niveau du pair central et ne permettent pas ainsi ni le passage à l’échelle ni la tolérance aux pannes. Ceci a donné naissance à d’autres solutions qui ont décentralisé le contrôle en s’appuyant sur les systèmes pair-à-pair : (i) les systèmes pair-à-pair non-structurés, (ii) les systèmes pair-à-pair structurés (iii) et les systèmes

hybrides. Les systèmes pair-à-pair non-structurés ont permis d'atteindre un degré d'autonomie important grâce à leur architecture décentralisée. Grâce à cette autonomie, ces systèmes permettent une maintenance du système faible due à la connexion ou déconnexion d'un pair. Cependant, il y a un risque fort d'avoir des résultats erronés ou incomplets. De plus, le mécanisme de diffusion de ce type de systèmes est celui par inondation, ce qui empêche le passage à l'échelle à cause du risque de saturation du réseau. Quant aux systèmes pair-à-pair structurés, ils ont montré une meilleure efficacité par rapport au mécanisme de recherche et du passage à l'échelle. Toutefois, maintenir leur structure rigide à jour demeure toujours un problème délicat face à la dynamicité des pairs dans de tels environnements, surtout en présence de l'effet 'Churn' (départ / arrivée de pairs de façon continue dans le système). Concernant les systèmes hybrides, ils évitent partiellement le problème de diffusion. L'efficacité de la méthode de découverte de sources de données dépend dans ce cas là uniquement de la façon dont les super-pairs communiquent entre eux. Néanmoins, ces systèmes risquent la saturation du super-pair puisqu'ils forment des goulets d'étranglement. De plus, ils sont aussi moins tolérants aux pannes dans le cas où un super-pair quitte le système.

Tous ces systèmes précédents ne prennent pas en compte l'aspect sémantique. Ainsi, d'autres solutions ont été proposées. Une première solution était d'établir des correspondances entre mots clés. Cependant, la définition de ces correspondances et leur mise à jour sont ardues. D'autres chercheurs ont opté alors pour l'utilisation d'un schéma global ou d'une ontologie globale. Néanmoins, la construction d'un tel schéma global ou ontologie globale est une tâche difficilement réalisable. C'est pourquoi plusieurs travaux ont opté pour l'utilisation de plusieurs ontologies de domaines. Cependant, toutes les solutions proposées imposent une topologie particulière pour les liens de 'mapping'. Ceci contraint fortement le système. Nous pouvons constater qu'il existe de manière publique des ontologies de domaine ainsi que les 'mappings' permettant la traduction. Ces ontologies et ces 'mapping' forment un graphe où chaque pair représente une ontologie et chaque arête un lien de 'mapping'. Ces

graphes n’ont pas de topologie particulière. Ainsi, aucune des méthodes présentées précédemment ne permettent de traiter ce type de graphe. En conséquence, dans le chapitre suivant, nous proposons une méthode de découverte de sources de données s’adaptant à tout type de topologie de graphe, si ce graphe est connexe. De plus, nous tacherons à proposer une méthode de recherche efficace limitant an maximum les coûts de maintenance surtout en présence de l’effet ‘Churn’.

Chapitre 3 : Méthode de découverte de sources de données prenant en compte la sémantique en environnement de grille

Chapitre 3 : Méthode de découverte de sources de données prenant en compte la sémantique en environnement de grille

1. Introduction
2. Découverte de sources de données prenant en compte l'aspect sémantique
   2.1. Architecture du système
   2.2. Méthode de découverte de sources de données
      2.2.1. Métadonnées décrivant les sources de données
      2.2.2. Découverte de sources de données intra-ontologie de domaine
      2.2.3. Découverte de sources de données inter-ontologies de domaines
3. Maintenance du système d'adressage
   3.1. Connexion d'un pair
   3.2. Déconnexion d'un pair
4. Conclusion

Méthode de découverte de sources de données prenant en compte la sémantique dans un environnement de grille

1. Introduction

La découverte de sources de données constitue une étape importante pour l'évaluation de requêtes dans un environnement à grande échelle. Cet environnement est caractérisé par (i) un grand nombre de sources de données, (ii) la dynamicité des pairs et (iii) l'hétérogénéité de ces sources de données.

Comme nous l'avons constaté dans le chapitre précédent, les premiers travaux de recherche concernant la découverte de sources de données étaient basés sur la recherche par mots clés. Les solutions basées sur les systèmes centralisés n'offrent pas de possibilité de passage à l'échelle à cause du goulet d'étranglement au niveau du serveur central. Les systèmes pair-à-pair (non-structurés, structurés ou hybrides) ont de bonnes propriétés par rapport au passage à l'échelle. Chaque type de système a ces avantages et inconvénients. Par exemple, en dépit de la fiabilité de la recherche dans les systèmes pair-à-pair structurés, les coûts de maintenance due à la dynamicité des pairs sont importants. Néanmoins, toutes les méthodes proposées ne prennent pas en compte les problèmes liés à la sémantique. Plusieurs travaux ont cherché à introduire la prise en compte de l'hétérogénéité dans la découverte de sources de données. La plupart s'est intéressée à l'hétérogénéité de schémas sans tenir compte de la sémantique. Par la suite, d'autres chercheurs ont introduit la sémantique dans la découverte de sources de données. Trois approches ont été proposées [Ket11a]: (i) l'approche basée sur la correspondance entre les noms utilisés dans les schémas [Sio05], [Ban04] et [Li05], (ii) l'approche basée sur l'utilisation d'une ontologie globale [Cru04], [Akb04] et [Haa04] et (iii) l'approche basée sur l'utilisation de plusieurs ontologies de domaine [Alk08],

[Hei04], [Zhu04a] et [Ket11b]. Nous avons constaté dans le chapitre précédent que l'approche basée sur l'utilisation de plusieurs ontologies de domaines est la plus viable. Toutefois, toutes les solutions proposées dans le cadre de cette approche imposent une certaine topologie de 'mapping' à respecter. On citera la topologie deux à deux [Hal03], les tables de 'mapping' [Are03, Gia05] et la topologie Super Pair [Akb04, Cru04b, Sar04, Fay06, Fay07]. Cela oblige l'administrateur à suivre cette topologie et créer ainsi les 'mappings' respectant cette dernière. Cette tâche est ardue et fastidieuse.

Une méthode de découverte de sources de données doit être efficace non seulement par rapport à la recherche tout en respectant l'aspect sémantique mais aussi par rapport à la mise à jour du système. De plus, elle doit s'adapter à n'importe quelle topologie de 'mapping' entre les ontologies. En effet, il existe sur Internet des milliers d'ontologies de domaines avec des liens de 'mappings' entre elles. Un challenge consiste donc à les réutiliser. Dans ce contexte, nous proposons une méthode de découverte de sources de données [Ket11b] s'adaptant à tout type de topologies de 'mapping' entre ontologies de domaines. De plus, notre méthode que nous décrivons dans la section suivante, prend en compte l'hétérogénéité sémantique des sources de données ainsi que la dynamicité des pairs.

2. Découverte de sources prenant en compte l'aspect sémantique

Dans cette section, nous présentons notre méthode de découverte de sources de données [Ket11b] dans un environnement à grande échelle. Un tel environnement est caractérisé par la présence d'un très grand nombre de sources de données qui de plus, sont hétérogènes et constamment en évolution. Généralement, la gestion de données dans un tel environnement s'appuie sur un schéma de placement. Cependant, nous ne pouvons pas disposer d'un schéma global centralisé puisqu'un tel schéma génère automatiquement un goulet d'étranglement dans un environnement à grande échelle. La duplication d'un tel schéma entrainera également des problèmes de mises à jour lors de la

modification (mise à jour) de ce schéma. La solution la plus adéquate consiste alors à la distribution d'un tel schéma. De plus, l'instabilité des pairs dans un tel environnement constitue un problème majeur du fait des connexions / déconnexions de manière continue de ces pairs. Ceci génère des coûts élevés de maintenance du système. Dans cette section, nous nous intéressons à la découverte de métadonnées décrivant des sources de données tout en prenant en compte l'aspect sémantique et l'instabilité des pairs dans de tels environnements.

2.1. Architecture du système

Concevoir une ontologie globale reste une tâche difficile à réaliser, vu la diversité des domaines dans un environnement tel que l'environnement biomédical. En effet, chaque domaine accumule de grandes masses de sources de données hétérogènes. Une solution consiste à décomposer un tel environnement en domaines. Ainsi, nous associons une ontologie de domaine [Nav05] à chaque domaine. Généralement, une grille de données peut être vue comme un ensemble d'organisations virtuelles (OV). Pour cela, nous associons une OV à chaque domaine. Ceci permet de prendre en compte le principe de localité [Har03] qui favorise l'autonomie de chaque OV. En général, les utilisateurs d'une grille de données accèdent souvent à des sources de données dans leur domaine d'application. Cela favorise les utilisateurs d'une OV qui accèdent souvent à des sources de données dans cette même OV.

Pour des raisons d'efficacité du processus de la découverte, les pairs au sein d'une même OV sont connectés suivant une Table de Hachage Distribuée (THD) [Sto01] comme le montre la Figure 1. La découverte de sources de données au sein d'une même ontologie de domaine revient alors à une découverte classique dans une THD. Ensuite, les OVs sont connectées entre elles de la manière suivante : on dit que les organisations virtuelles OV_i et OV_j sont connectées si et seulement s'il existe une relation de 'mapping' entre les ontologies associées à OV_i et OV_j respectivement. Il reste à définir la façon dont les OVs communiquent entre elles. Nous nous appuyons sur les relations

(règles) de ‘mapping’ existantes entre ontologies de domaines différentes. Puis, nous définissons un système d’adressage pour relier les organisations virtuelles entre elles.

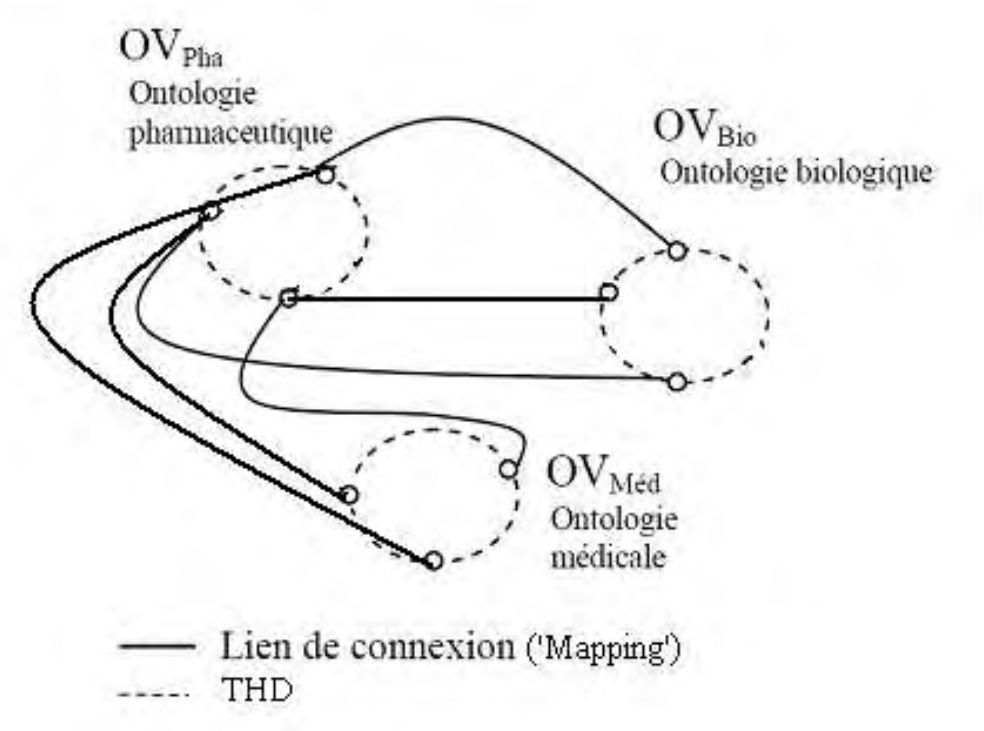


Figure 1 : Exemple d’associations ontologies de domaines / organisations virtuelles

Soit $N(OV_i)$ l’ensemble des organisations virtuelles connectées à OV_i via des relations de ‘mappings’ (i.e. OVs voisines de OV_i). Notons par $|N(OV_i)|$ le nombre d’organisations virtuelles voisines de OV_i. L’ensemble des OVs et les liens présentant les ‘mappings’ entre ces OVs forment un graphe noté G(N, E) non-orienté avec N l’ensemble des pairs présentant les OVs et E l’ensemble d’arêtes présentant des liens de ‘mapping’ entre les OVs. Notons que le lien e_{ij} ϵ E si et seulement si il existe une relation de ‘mapping’ entre OV_i et OV_j. Pour assurer le fait que tous les domaines peuvent être atteints par une requête de découverte de source de données émise à partir de n’importe quel pair, le graphe de ‘mappings’ doit être connexe. Ainsi, nous supposons, dans le reste

de ce chapitre, que le graphe est connexe. En conséquence, pour tout $OV_i \in N$ et $OV_j \in N$, il existe un chemin $c_{ij} \in E$ allant de OV_i à OV_j.

Pour chaque pair P_k de OV_i, nous associons $|N(OV_i)|$ points de sortie PS_k. Ces points de sortie appartiennent à l'ensemble $N(OV_i)$. Notons que deux pairs appartenant à une OV_i doivent éviter de référencer les mêmes pairs dans $OV_j \in N(OV_i)$. En effet, si un pair $\in OV_j$ est référencé par plusieurs points de sorties $\in OV_i$, celui-ci peut constituer un goulet d'étranglement au niveau de OV_i. Aussi, ce pair constituerai également un point unique de défaillance 'single point of failure'. Ceci est d'autant plus indispensable lorsqu'on évoque l'instabilité de notre environnement.

2.2. Méthode de découverte de sources de données

Après avoir décrit la façon dont les OVs sont connectées entre elles, cette section décrit le processus de découverte de sources de données. Nous classons les requêtes de découverte de sources de données en deux types :

(i) Les requêtes au sein d'une même ontologie de domaines appelées les requêtes de découverte de sources de données intra-ontologie de domaine. Elles sont basées sur le système de routage classique dans une table de hachage distribuée (THD) [Sto01].

(ii) Les requêtes entre ontologies de domaines appelées les requêtes de découverte de sources de données inter-ontologies de domaines. Pour évaluer ce type de requêtes, nous utilisons le système d'adressage, défini précédemment, pour relier les OVs les unes aux autres.

2.2.1. Métadonnées décrivant les sources de données

Afin d'illustrer le processus de découverte de ressource par notre méthode, nous considérons la requête SQL suivante; *Select * From Doctors*. Le processus de découverte de source de données consiste à découvrir les métadonnées décrivant la relation *Doctors* associée à un concept décrit dans l'ontologie de domaine correspondante. Ces métadonnées comportent entre autres: (i) le profil de la relation *Doctors* (e.g., attributs, clé), (ii) le placement

de la relation *Doctors* (e.g., adresse IP de chaque fragment, fragmentation, information sur la duplication ainsi que la règle de reconstruction de la relation *Doctors*) et (iii) les différentes estimations de la relation *Doctors*, comme la taille, utilisées durant l'optimisation de la requête concernée.

2.2.2. Découverte de sources de données intra-ontologie de domaine

Comme nous l'avons signalé précédemment, les requêtes intra-ontologie de domaine sont évaluées suivant le système de routage utilisé dans une THD classique [Sto01]. En effet, les systèmes pair-à-pair structurés, adaptés dans notre méthode, ont prouvé leur efficacité par rapport au passage à l'échelle ainsi que pour l'efficacité de recherche. Ainsi, une fonction de hachage est appliquée. La clé étant le nom du concept à découvrir (e.g., la relation *Doctors* dans notre exemple précédent). Si le pair émetteur de la requête de découverte de sources de données et le pair responsable de cette source de données appartiennent à la même ontologie de domaine, il s'agit alors d'une requête intra-ontologie de domaine (Figure 2). Rappelons que la complexité pour trouver le pair responsable de la source de données à découvrir est de l'ordre de O(Log(N)) où N représente le nombre de pairs dans cette OV [Sto01]. Le pair responsable d'une source de données envoie uniquement les métadonnées de cette source de données. Le résultat de cette découverte étant les métadonnées décrivant ce concept.

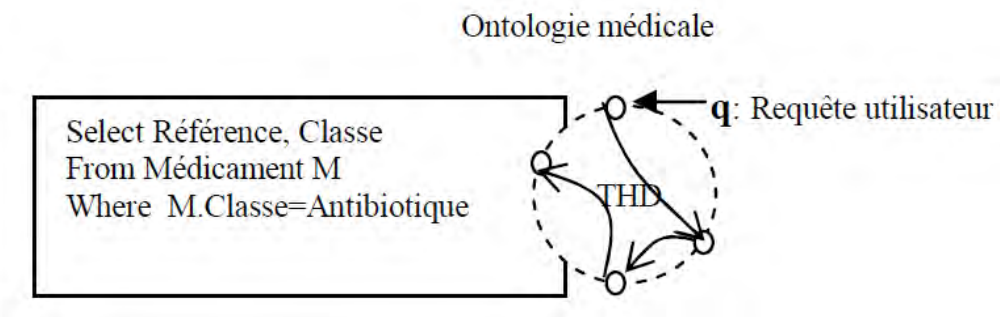


Figure 2 : Exemple de découverte de sources de données intra-ontologie de domaine

2.2.3. Découverte de sources de données inter-ontologies de domaines

Afin de décrire le processus de découverte de source de données inter-ontologies de domaines, supposons que P_k soit le pair recevant la requête *q* soumise par un utilisateur appartenant à l'organisation virtuelle OV_i associée à l'ontologie de domaine O_i. Supposons également que C est le concept à découvrir via la requête de découverte de ressource. Supposons enfin que P_j est le pair responsable de C', le concept correspondant à C dans l'ontologie de domaine O_j avec $i \neq j$.

Pour aboutir au résultat final de la découverte de sources de données et avoir une réponse fiable, la recherche doit être effectuée non seulement dans le domaine local mais aussi dans toutes les ontologies de domaines voisines. Ceci nécessite alors d'utiliser des points de sortie référençant des pairs dans ces ontologies de domaine. Nous nous basons sur les 'mappings' existants entre ces ontologies pour traduire le concept à découvrir à travers les liens de 'mappings' existants entre les ontologies de domaines. En effet, le concept C est traduit suivant les règles de 'mappings' existantes entre les différentes ontologies de domaines traversées lors de la découverte de source de données. La figure 3 montre un exemple d'acheminement de requêtes de découverte inter ontologie entre O_i et O_j à travers le point de sortie $PS_{k+1,j}$ et un autre exemple requête de découverte inter-ontologie entre O_i et O_{j+2} à travers le point de sortie $PS_{k+2,j}$.

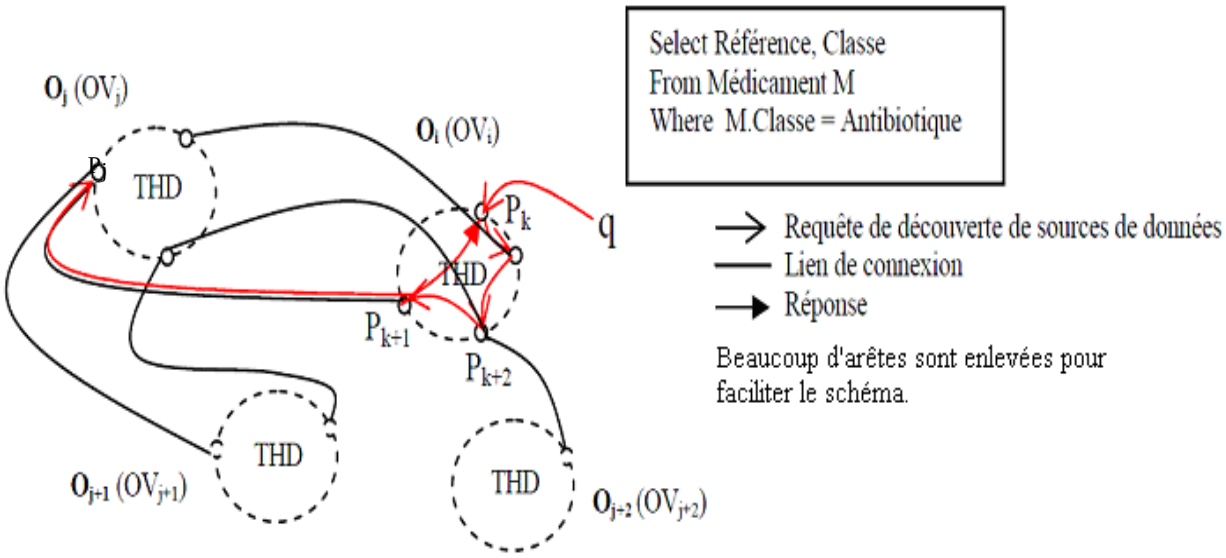


Figure 3 : Exemple de découverte de sources de données inter-ontologies de domaines

La figure 4 montre l'algorithme de découverte de sources de données. Ce dernier décrit le processus de découverte intra et inter-ontologies de domaines. Initialement la requête de découverte est appelée via une fonction *Lookup* ayant pour paramètres (i) le concept recherché, (ii) l'OV dans laquelle ce concept est recherché, (iii) le pair initiateur de la requête et (iv) le chemin suivi par la requête durant le processus de traduction du concept C. Ce chemin étant en paramètre de sortie de cette fonction. Pour simplifier cet algorithme, nous n'avons pas décrit la mise à jour des points de sortie dans le cas d'une déconnexion de pairs. Ainsi, à chaque fois qu'une requête de découverte de sources de données est reçue au niveau d'une OV_i, on recherche tous les points de sorties vers les autres OV_j avec $i \neq j$. Si un pair $P_k \in OV_i$ constate que son point de sortie $PS_{k,j}$ vers une OV_j distante ne répond pas alors il contacte son voisin qui a un autre point de sortie $PS_{k+1,j}$ vers cette même OV_j. Ceci est répété jusqu'à obtenir un point de sortie connecté dans OV_j. Quand il réussit à trouver un point de sortie vers OVj, le pair en profite pour mettre à jour son point de sortie. Afin d'éviter d'avoir un point de sortie identique à P_{k+1}, il demande au point de sortie son plus proche voisin. Ce plus proche voisin

deviendra alors son nouveau point de sortie. Ce processus est le même quand il s'agit de l'arrivée d'un nouveau pair. Nous discutons ce cas dans la section réservée à la connexion / déconnexion d'un pair en utilisant notre méthode. Dans le cas ou tous les voisins contactés répondent par la négative, l'organisation virtuelle OV_j est considérée comme étant déconnectée. Seulement l'administrateur pourra procéder à la mise à jour d'un point de sortie vers cette OV. Signalons aussi qu'afin d'éviter l'attente infinie d'une réponse au niveau d'un point de sortie, nous définissons un intervalle de temps; RTT (Round-Trip-Time). Ainsi, si un point de sortie ne reçoit pas de réponse au bout d'un RTT, le pair qu'il référence est supposé déconnecté.

```
//G(N, E) : Graphe non-orienté connexe.
//N : Ensemble de pairs présentant les OVs.
//E : Ensemble des arêtes présentant les liens de 'mapping'
//entre les OVs.
//PS : Ensemble des points de sortie d'un pair Pk ∈ OVi. Cet
//ensemble est constitué d'un pair pour chaque OVj ou il
//existe une arête eij dans G reliant OVi à OVj.
//PSk,j : Point de sortie d'un pair Pk ∈ OVi référençant un
//pair ∈ OVj i!=j.
//Chemin : Chemin suivi lors de la découverte de sources de
//données.
//Lookup(C, OVj, Pk, Chemin) : Découvrir le concept C
// à travers le pair PSk,j.
//TTL : 'Time-To-Live' (limite de la propagation du message).

Metadata ← Lookup(C, OVPSk,j, Pk, Chemin);
//Requête de découverte Intra-ontologie de domaine.
TTL ← TTL–1;
If(TTL!=0) then
For each PSk,j ∈ PS
Metadata ← Metadata U Lookup(Traduit(C, OVj, OVPSk,j),
OVPSk,j, Pk, Chemin U OVj);
//Requête de découverte Inter-ontologies de domaines.
If(not Empty(Metadata))
  then Return(Metadata, Pk, Chemin U OVj);
```

Figure 4 : Algorithme de découverte de sources de données inter-ontologies de domaines

Nous définissons également un niveau de propagation inter-ontologie de domaine. Ainsi, une ontologie de domaine O_{j+1} ne possédant pas de liens de 'mapping' avec O_i mais reliée à O_j peut être atteinte à travers la propagation de la requête d'O_j vers O_{j+1}. Afin d'éviter une propagation infinie de la requête de

découverte, un paramètre TTL (Time To Live) est défini. Il permet de définir le nombre maximum de propagation entre ontologies de domaines. Pour assurer une bonne complétude du résultat, ce dernier doit être défini par le diamètre du graphe (i.e., longueur du chemin le plus long). Enfin, lorsque le concept recherché est trouvé, les métadonnées décrivant le concept C' sont envoyés au pair P_k initiateur de la requête q. La réponse contient également le chemin suivi par la requête durant le processus de traduction du concept C d'une ontologie de domaine vers une autre pour atteindre le pair responsable du concept C'.

Exemple 1

L'exemple suivant décrit le processus de découverte de ressources inter-ontologie de domaine. Considérons quatre organisations virtuelles : OV_{Bio} : représentant l'OV associée au domaine biologique, $OV_{Méd}$: représentant l'OV associée au domaine médical, OV_{Pha}: représentant l'OV associée au domaine pharmacologique et $OV_{Gén}$: représentant l'OV associée au domaine génétique (Figure 5). Supposons que le pair $N_{11} \in VO_{Bio}$ émet une requête de découverte de sources de données afin de rechercher les métadonnées relatives au concept *Médicament*.

En plus du processus intra-ontologie de domaine, la requête de découverte de sources de données est aussi envoyée à l'ensemble des OVs connectés à OV_{Bio} via les points de sorties permettant la liaison avec ces OVs. Celles-ci correspondent aux domaines Pharmaceutique et Génétique dans notre exemple en Figure 5. Le pair N_{11} commence par tester ses connexions avec les pairs qu'il référence dans les autres OVS. A chaque fois qu'il ne reçoit pas de réponse d'un pair qu'il référence, il contacte son voisin puis le voisin de son voisin et ainsi de façon récursive jusqu'à ce qu'il trouve un point de sortie connecté dans les autres OVs. Dans notre exemple, il trouve le pair N_{14} de son OV qui référence le point de sortie N_{23} permettant l'accès à OV_{Pha}. Le même processus est répété avec la redirection de la requête vers $OV_{Gén}$ à travers le point de sortie N_{12}. Rappelons également que si le niveau de propagation dépasse 1, le pair N11 peut atteindre l'$OV_{Méd}$ à travers les liens de connexion

existants entre OV_{Pha} et $OV_{Méd}$. Finalement, les métadonnées décrivant le concept recherché sont envoyées à N_{11} via le chemin inverse. Ces métadonnées contiennent notamment le chemin parcouru durant la requête de découverte de sources de données.

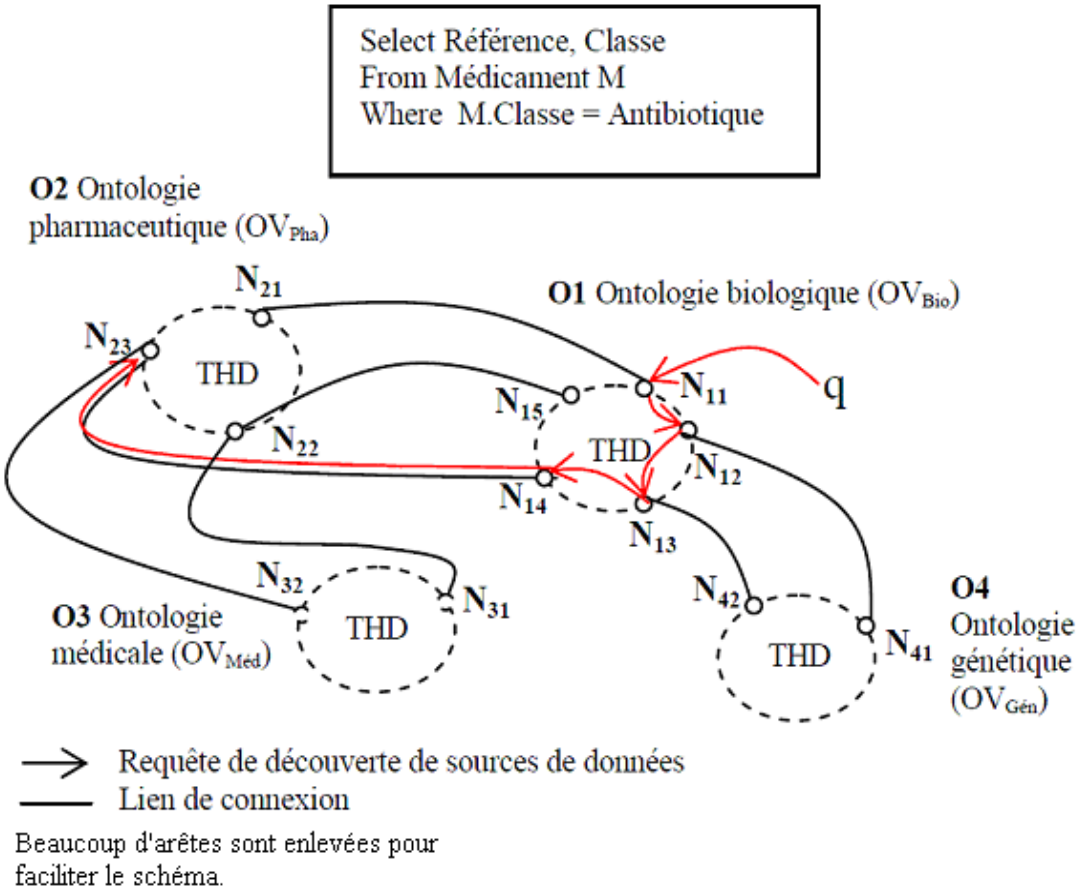


Figure 5 : Exemple de processus de découverte de sources de données inter-ontologies de domaines

3. Maintenance du système d'adressage

La dynamicité des pairs constitue une caractéristique d'un environnement à grande échelle telle que la grille. En effet, chaque pair peut rejoindre ou quitter le système à tout moment. Cela pose des problèmes de mises à jour au niveau du système. Dans les systèmes pair-à-pair structurés tel que Chord [Sto01], utilisé dans notre système, la mise à jour de la THD engendre $Log^2(N)$ messages pour chaque connexion / déconnexion d'un pair (N constitue le

nombre total des pairs du système). Rappelons que la connexion / déconnexion peut concerner simultanément un ou plusieurs pairs du système ce qui augmente le nombre de messages nécessaires à la maintenance d'un tel système surtout avec la présence d'un état 'Churn'. En effet, 'Churn' correspond à la connexion / déconnexion des pairs de manière continue. Dans cette section, nous proposons une méthode qui permet de réduire de façon significative les coûts de maintenance du système. Dans ce contexte, nous distinguons deux types de maintenance : (i) la maintenance de la THD associée à une OV_i et (ii) la maintenance des pairs points de sortie référençant des pairs appartenant à $N(OV_i)$ avec $i \neq j$.

Dans cette section, nous ne discutons pas la maintenance d'une THD. En effet, ceci revient à une maintenance classique d'une THD [Sto01]. Cela survient lors de la connexion / déconnexion d'un pair au sein d'une OV. Par contre, la maintenance des pairs 'points de sortie' pose des problèmes de mises à jour plus complexes. Elle consiste à mettre à jour les références de ces points de sorties lors de chaque connexion et déconnexion d'un pair. Ces coûts de maintenance sont d'autant plus importants lors de la déconnexion d'un pair. En effet, la déconnexion ou la panne d'un pair point de sortie a pour conséquence l'impossibilité d'accès à l'OV distante. La maintenance du système, dans ce cas, revient à maintenir les références du pair 'point de sortie' concerné par la déconnexion ou la panne. Dans notre cas, les pairs qui référencent ce point ne sont pas informés par cette déconnexion et la mise à jour se fait lors du processus de découverte de ressource. On parle alors d'une mise à jour 'paresseuse'. Dans ce qui suit, nous étudions la mise à jour du système lors de la connexion et de la déconnexion d'un pair.

3.1. Connexion d'un pair

Supposons qu'un nouveau pair *NouvP* se connecte à une OV_i. La première étape consiste à trouver tous les points de sorties vers toutes les autres OVs appartenant à $N(OV_i)$. Pour cela, il contacte le voisin le plus proche dans OV_i pour avoir ses points de sortie vers les autres OVs appartenant à $N(OV_i)$.

Chacun de ces points de sorties est alors contacté. Si au bout d'une certaine période de temps (RTT), un point de sortie référençant un pair dans OVj ($i \neq j$) ne répond pas, il est considéré comme étant déconnecté. Le pair NouvP contacte alors le voisin de son voisin (à travers la fonction *Voisin* dans la figure 6) et réitère cela (à travers la fonction *Suivant*) jusqu'à trouver un point de sortie dans OV_j. Ceci est répété afin d'établir tous les points de sorties vers toutes les OVs appartenant à $N(OV_i)$. Le test est établi par la fonction *Verifie* dans l'algorithme décrivant la connexion d'un pair dans la figure 6. Finalement, le nouveau pair NouvP envoie un message au pair point de sortie trouvé afin que ce dernier lui donne son voisin dans l'OV_j distante. Il référencera alors ce nouveau pair dans OV_j. Ceci permet d'avoir au moins deux points de sortie différents dans OV_i qui ne référencent pas les mêmes pairs dans OV_j ($i \neq j$). Nous évitons ainsi de former un seul point d'accès. Dans le cas où aucune réponse provenant de tous les voisins contactés n'est positive, l'OV_j est considérée comme déconnectée.

```
//Cet algorithme se déclenche après la connexion d'un
//pair G(N, E) : Graphe non-orienté connexe.
//N : Ensemble de pairs présentant les OVs.
//E : Ensemble des arêtes présentant les liens de
//'mapping' entre les OVs.
//PS : Ensemble des points de sortie d'un pair Pk ∈
//OVi. Cet ensemble est constitué d'un pair pour
//chaque OVj ou il existe une arête eij dans G reliant
//OVi à OVj.
//Verifie(PSk,j) : Vérifier la connexion du point de
//sortie Pk vers OVj.
//Trouve == TRUE: Point de sortie connecté dans OVj
//après le test de Vérification.

Pour j ∈ [1, |N(OVi)|]
{
Trouve ← Faux;
Suivant = NouvP.Voisin;
 Tant que(Suivant != NouvP) && (Trouve == Faux)
 {
 Trouve ← Verifie(PSk,j → Suivant());
 Si(Trouve) alors PSk,j = PSk,j → Suivant();
 // NouvP reference PSk,j → Suivant()
 Sinon Suivant = Voisin(Suivant);
 }
}
```

Figure 6 : Algorithme de connexion d'un nouveau pair.

La procédure de connexion, décrite dans l'algorithme de la figure 6 est exécutée lors de la connexion d'un pair au système.

Exemple 2

Supposons les quatre organisations virtuelles : OV_{Bio} représentant l'OV Biologique, OV_{Pha} représentant l'OV Pharmaceutique, OV_{Med} représentant l'OV Médicale et OV_{Gen} représentant l'OV Génétique.

Supposons qu'un pair N_{11} se connecte à OV_{Bio} (biologie). La première phase de la maintenance consiste à une maintenance classique de la DHT. Puis, ce pair tente d'établir des points de sorties vers $OV_{Pha} \in N(OV_{Bio})$. Il contacte d'abord le pair N_{12} qui référence le pair N_{24} dans OV_{Pha}. Cependant, supposons que le pair référencé N_{24} ne répond pas après un certain temps RTT. Il est considéré alors comme étant déconnecté. N_{11} contacte alors le voisin de N_{12} puis les voisins de son voisin jusqu'à ce qu'il trouve un point de sortie qui référence un pair dans OV_{Pha}. Il trouve finalement que N_{14} qui référence le pair N_{23} qui est connecté dans OV_{Pha}. Ce dernier pair contacte son voisin N_{22} qui sera dorénavant référencé par le pair N_{11}. La même procédure est suivie pour la mise à jour des points de sorties vers OV_{Gen}. Ainsi, le pair N_{11} référence N_{43} dans OV_{Gen}. Enfin, dans le cas ou tous les pairs référencés par les voisins de N_{11} ne référencent pas de pairs connectés, OV_{Pha} est considérée comme déconnectée (Voir figure 7).

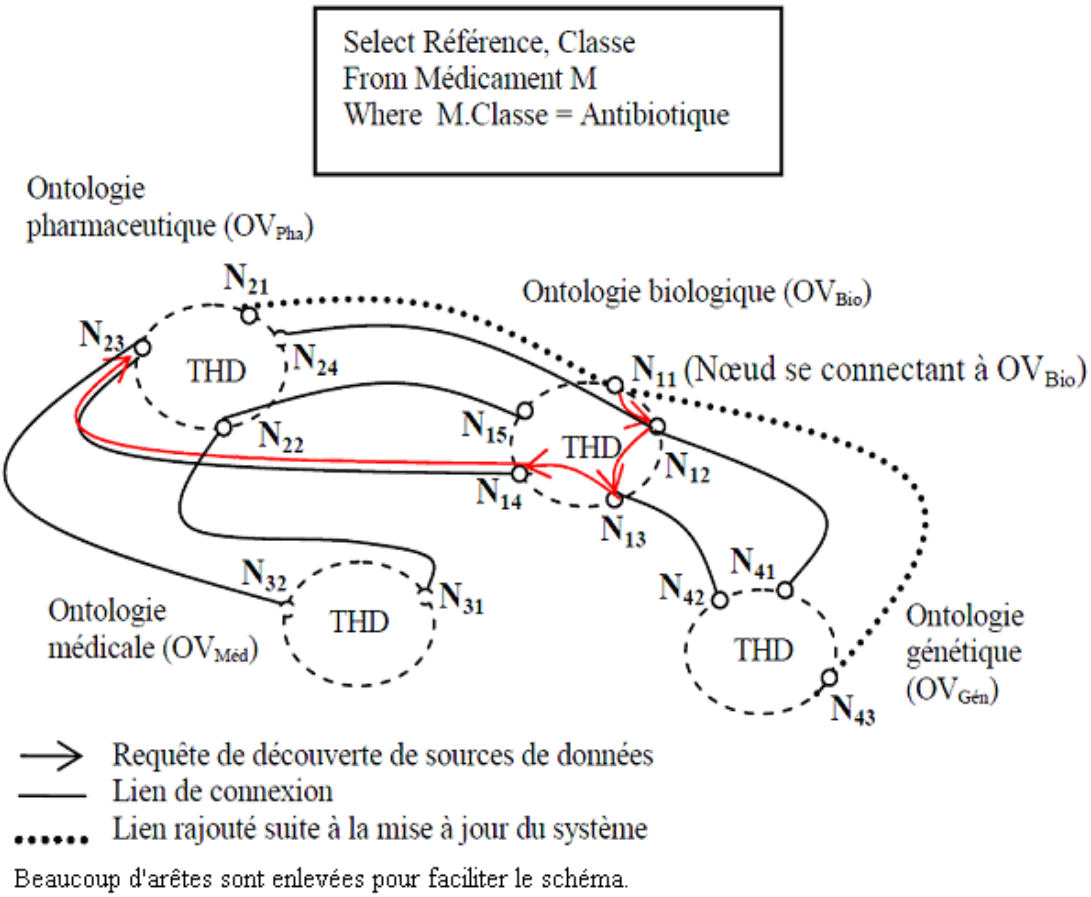


Figure 7 : Exemple de connexion d'un nouveau pair.

3.2. Déconnexion d'un pair

La déconnexion d'un pair d'une OV entraine deux types de mises à jour. Dans un premier temps, une mise à jour classique de la THD est nécessaire. Ensuite, la mise à jour des points de sorties est effectuée. En effet, tous les pairs qui référencent ces points de sorties doivent être mis à jour. Cette mise à jour engendre un coût de maintenance supplémentaire non négligeable surtout en présence d'un effet 'Churn'. Ce coût peut devenir alors prohibitif. Une première solution consiste à informer les pairs qui référencent dans les OV distantes de cette déconnexion.

Dans notre méthode, nous ne procédons pas de cette manière. Afin d'éviter ces coûts de maintenance, nous proposons une maintenance dite 'paresseuse'. En effet, lors de la déconnexion d'un pair, les pairs qui référencent dans les autres OVs connectées ne sont pas informés. La maintenance est faite lors du processus de découverte de sources de données.

Afin de mieux comprendre ce processus, supposons la déconnexion d'un pair *NDec* ϵ OV_i. En premier lieu, une mise à jour classique de la THD est effectuée. En plus, il faut mettre à jour les points de sorties référençant ce pair. En effet, si le pair déconnecté constitue un point de sortie d'un pair à une OV_k ϵ $N(OV_i)$, il faut mettre à jour toutes les références liées à ce pair. Deux solutions émergent. Une première solution consiste à propager l'information à toute les OV ϵ $N(OV_i)$, vers lesquelles *NDec* est connecté. Cette stratégie utilise le procédé de l'inondation au niveau des OVs. Nous ne l'adoptons pas. Une autre solution, adoptée dans notre système, consiste à effectuer une maintenance paresseuse. Les pairs qui référencent ne sont pas avertis de la déconnexion de NDec. En conséquence, NDec ne contacte pas les pairs qui le référencent dans OV_k ϵ $N(OV_i)$. Les pairs qui référencent par *NDec* seront mis à jour lors du processus de découverte de sources de données inter-ontologies de domaines. Ceci nous permet de réduire le nombre de messages de maintenance ce qui réduit le coût de la maintenance et ce qui permet d'éviter un coût de maintenance prohibitif. Notons que ce mécanisme de mise à jour paresseuse est efficace (système mis à jour de manière continue) si le processus de découverte est utilisé fréquemment.

4. Conclusion

Dans ce chapitre, nous avons proposé une méthode de découverte de sources de données dans un environnement à grande échelle en prenant en compte la sémantique ainsi que la dynamicité des pairs. Dans notre méthode, une ontologie de domaine est associée à chaque domaine dont les pairs constituent une organisation virtuelle (OV). Afin d'assurer la connexion entre les OVs connectées entre elles, nous avons proposé un système d'adressage

permettant un accès permanent de n'importe quelle ontologie vers une autre. Notre unique hypothèse est que le graphe de 'mappings' entre ces ontologies soit connexe. Chaque pair d'une OV_i référence un point de sortie dans une autre OV_j. Ces pairs de sortie sont choisis en évitant qu'un pair appartenant à OV_j ne soit référencé par plusieurs pairs appartenant à OV_i. Ceci permet d'éviter des goulets d'étranglement lors de l'accès à une OV_j. La diffusion des requêtes de découverte de sources de données au sein d'une OV se fait suivant le routage classique utilisé dans une THD. On parle alors de requêtes intra-ontologie de domaine. Concernant les requêtes inter-ontologies de domaines, on se basera sur le système d'adressage mis au point à travers les points de sortie.

Notre méthode permet également de limiter le nombre des messages nécessaire à la maintenance du système d'adressage. Une mise à jour dite paresseuse est adoptée lors de la connexion / déconnexion de pairs. La mise à jour n'est pas faite au moment de la déconnection d'un pair mais lors du processus de découverte de sources de données. Cela permet une réduction significative des coûts de maintenance surtout en présence d'un effet 'Churn'. En effet, dans beaucoup de systèmes, les coûts de mises à jour en cas d'effet 'Churn' sont importants et rendent le système inutilisable.

Dans le chapitre suivant, nous évaluons les performances de notre méthode afin d'étudier son efficacité ainsi que sa viabilité.

Chapitre 4 : Évaluation des performances

Chapitre 4 : Évaluation des performances

1. Introduction
2. Environnement de simulation
3. Découverte de sources de données inter-ontologies de domaines
   3.1. Découverte de sources de données inter-ontologies de domaines en utilisant une seule requête
   3.2. Découverte de sources de données inter-ontologies de domaines en utilisant plusieurs requêtes
4. Maintenance du système
5. Discussion et conclusion

Évaluation des performances

1. Introduction

Pour mesurer la viabilité de notre méthode nous réalisons une étude qualitative et quantitative. Dans cette section, nous évaluons les performances de la méthode proposée à travers une comparaison de celle-ci aux méthodes de découverte de sources de données suivantes: (i) la découverte de sources de données par inondation (Resource Discovery by Flooding Method) [Iam01, Kal02, Iam04, Abd05, Tal05, Jea08], (ii) la découverte de sources de données en s'appuyant sur un réseau de type super-pair (Resource Discovery by 'Super Peer Method') [Akb04, Cru04b, Sar04, Fay06, Fay07] et (iii) la découverte suivant une topologie de 'mapping' deux à deux (Resource Discovery by Pair Method) [Hal03]. Notre étude consiste à comparer les résultats de l'évaluation de notre méthode par rapport à ceux de ces trois dernières méthodes tout en prenant en compte l'hétérogénéité sémantique des sources de données biomédicales.

Dans un premier temps, nous présentons notre environnement de simulation ainsi que nos hypothèses. Ensuite, nous décrivons et analysons nos différents tests de simulation pour l'évaluation de notre méthode. Durant les sections qui suivent, nous nous intéressons notamment: (i) aux temps de découverte de sources de données inter-OVs vu que la découverte intra-OV se fait suivant un système d'adressage classique de DHT et (ii) à l'impact de la connexion / déconnection de pairs sur la maintenance du système. Finalement, nous concluons cette section d'évaluation des performances par une discussion.

2. Environnement de simulation

Pour évaluer les performances de notre méthode de découverte de sources de données à grande échelle, nous avons besoin d'un nombre important de pairs. Nous ne disposons pas de ce nombre de pairs ni de l'infrastructure réseau nécessaire. Ainsi, l'évaluation des performances sera effectuée par simulation.

Cette simulation permettra, d'abord, une évaluation des temps de réponse de la découverte de sources de données. Ce temps de réponse est le temps entre la soumission d'une requête de découverte et la réception de la réponse contenant l'ensemble des métadonnées associées à la source de données. Dans ce chapitre, le temps de réponse et le temps de découverte de sources de données désignent la même chose. Enfin, nous abordons la simulation de la maintenance du système lors de sa mise à jour due à une connexion ou une déconnection de pairs.

Lors de nos évaluations, nous avons utilisé Open-Chord [Ope04] pour simuler une DHT et le simulateur ns-2[8] pour simuler la découverte de sources de données inter-ontologies de domaines. Cela est fait pour les différentes topologies utilisées dans les quatre méthodes comparées. Sous ns-2, les OVs sont représentées sous formes de pairs et les 'mapping' liant ces OVs sont représentés par des arêtes. Nous supposons que le temps de communication (coût du saut) entre deux OVs (associées aux ontologies de domaines) est de 10ms (une constante pour tous les 'mapping' entre OVs).

3. Découverte de sources de données inter-ontologies de domaines

Cette section décrit l'évaluation de performances de la méthode proposée de découverte de sources de données. Plus précisément, nous évaluons les performances des requêtes inter-ontologies de domaines. Nous nous intéressons aux temps de réponses de la découverte pour chacune des méthodes en variant dans un premier temps le nombre d'OVs puis le nombre de requêtes soumises par seconde et par OV. Nous évaluons également les capacités de chacune de ces méthodes à maintenir le système dans le cas de connexion / déconnexion de pairs.

[8] Le simulateur de réseau ns-2, http://isi.edu/nsnam/ns/

Dans ces évaluations, nous respectons les topologies imposées dans chacune des méthodes implantées. Ainsi, dans la méthode basée sur le processus d'inondation que nous noterons M1 dans les courbes, le processus de découverte s'appuie sur une diffusion vers tous les pairs. Autrement dit, un mécanisme d'inondation est utilisé pour l'envoi de requêtes. Dans cette méthode, il n'y a pas vraiment de topologie de 'mapping' précise à respecter, mais la recherche s'effectue par mot clés. La sémantique au niveau de cette méthode intervient dans la recherche du concept et de ses synonymes mais pas dans le processus d'adressage et de communication entre les OVs. La deuxième méthode à laquelle nous avons comparé nos résultats, notée par M2 dans nos expériences, repose sur une topologie de 'mapping' basée sur les systèmes super-pair. Dans cette méthode, chaque OV dispose d'un seul point de sortie vers une autre OV. Pour la troisième méthode notée M3, la topologie de 'mapping' imposée est celle en anneau (deux à deux). Finalement, dans notre méthode que nous noterons M4, les OVs (ontologies de domaines) sont interconnectées entre elles suivant des liens de 'mapping' en n'imposant aucune topologie particulière. Notre unique hypothèse est que le graphe représentant nos OVs soit connexe pour pouvoir atteindre n'importe quelle OV dans le système à travers le système d'adressage mis au point.

Nous commençons d'abord par évaluer le processus de découverte. Nous nous intéressons au temps de réponse relatif au processus de la découverte de sources de données inter-OVs. Dans un premier temps, nous avons utilisé une seule requête de découverte de sources de données puis plusieurs requêtes par seconde tout en variant le nombre d'OVs. Nous avons fixé le nombre de pairs par OV à 10.

3.1. Évaluation des performances pour une seule requête

La figure 1 montre l'évolution du temps de réponse des requêtes inter-OVs lors de la variation du nombre d'OV de 1 à 50. Ceci nous renseigne sur le passage à l'échelle des méthodes de découverte de sources de données.

Précisons que dans ces expériences, une seule requête de découverte de concept est transmise par OV.

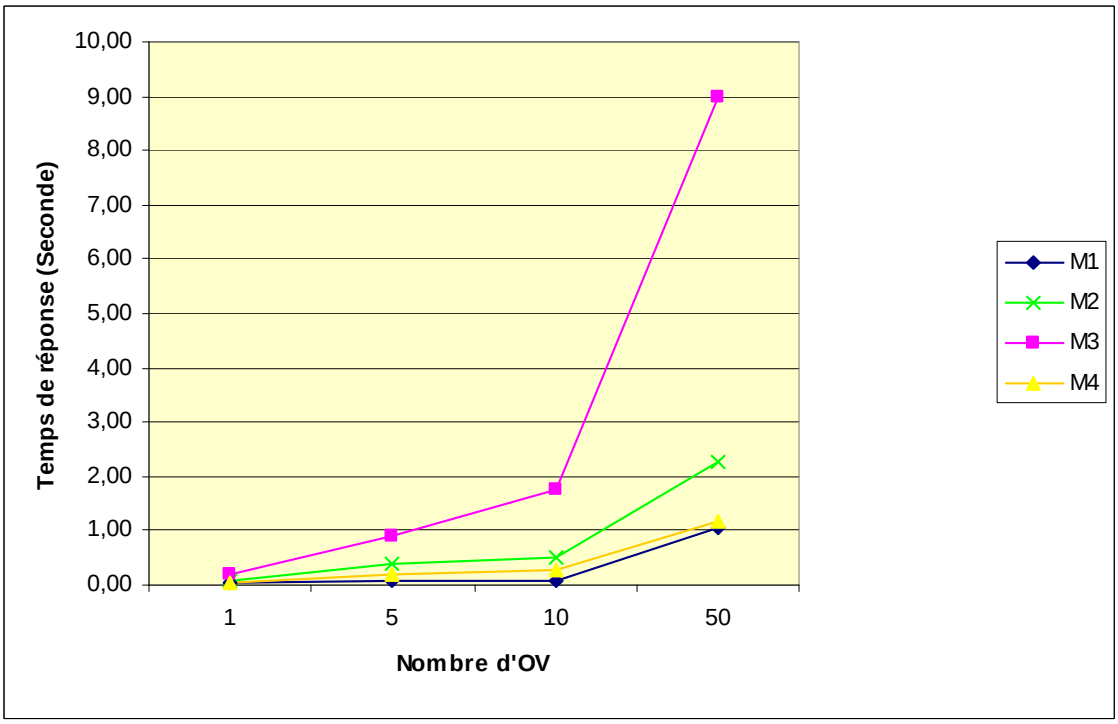


Figure 1 : Evolution du temps de réponse de la découverte de sources de données par rapport au nombre d'OV (pour une seule requête de découverte)

Nous pouvons constater un grand écart entre les temps de réponses des différentes méthodes comparées. Ainsi, le temps de réponse de la méthode M3 est le plus important par rapport aux trois autres méthodes. Ceci est dû à la longueur du chemin à traverser pour retrouver le concept à découvrir. Les temps de réponse de la méthode M2 sont aussi supérieurs au temps de réponse de notre méthode. Ceci provient du fait que dans notre méthode, il y a plusieurs pairs 'points de sorties' permettant d'interconnecter les OVs entre elles. En conséquence, il existe plusieurs liens possibles pour une requête de découverte de sources de données entre deux OVs. En utilisant la topologie basée sur les super pairs (méthode M2), on ne retrouve qu'un seul pair permettant la connexion entre deux OVs. En effet, envoyer une seule requête dans la méthode M2 peut provoquer une saturation au niveau d'un pair super-pair

voulant transmettre cette requête à d'autres pairs super-pairs. Ces messages sont alors sérialisés.

Comme le montre la Figure 1, les méthodes M1 et M4 présentent de meilleurs résultats en termes de temps de réponse par rapport au nombre d'OV existante. Néanmoins, la méthode M1 permet des temps de réponses meilleures que les nôtres. En utilisant notre méthode, on retrouve des temps de réponses supérieurs de 30 % en moyenne à ceux obtenus par la méthode M1. Toutefois, le processus de découverte de sources de données dans cette méthode se base sur la recherche par mot clés ce qui nuit à la précision et à la complétude du résultat. En effet, tous les résultats risquent de ne pas être trouvés malgré leur existence. De plus, des résultats erronés peuvent être trouvés. Dans notre méthode, nous ne rencontrons pas ce type de problème vu que le processus de découverte de sources de données utilise les 'mappings' et traduit les concepts d'une OV à une autre. Ceci permet, évidemment, plus de complétude et de précision dans les résultats des requêtes de découverte de sources de données.

Nous nous sommes également intéressés au facteur d'accélération entre ces temps de réponses en comparant les résultats obtenus par les quatre méthodes.

Les figures 2 et 3 montrent le facteur d'accélération des temps de réponses obtenus par notre méthode M4 par rapport aux autres méthodes M1, M2 et M3.

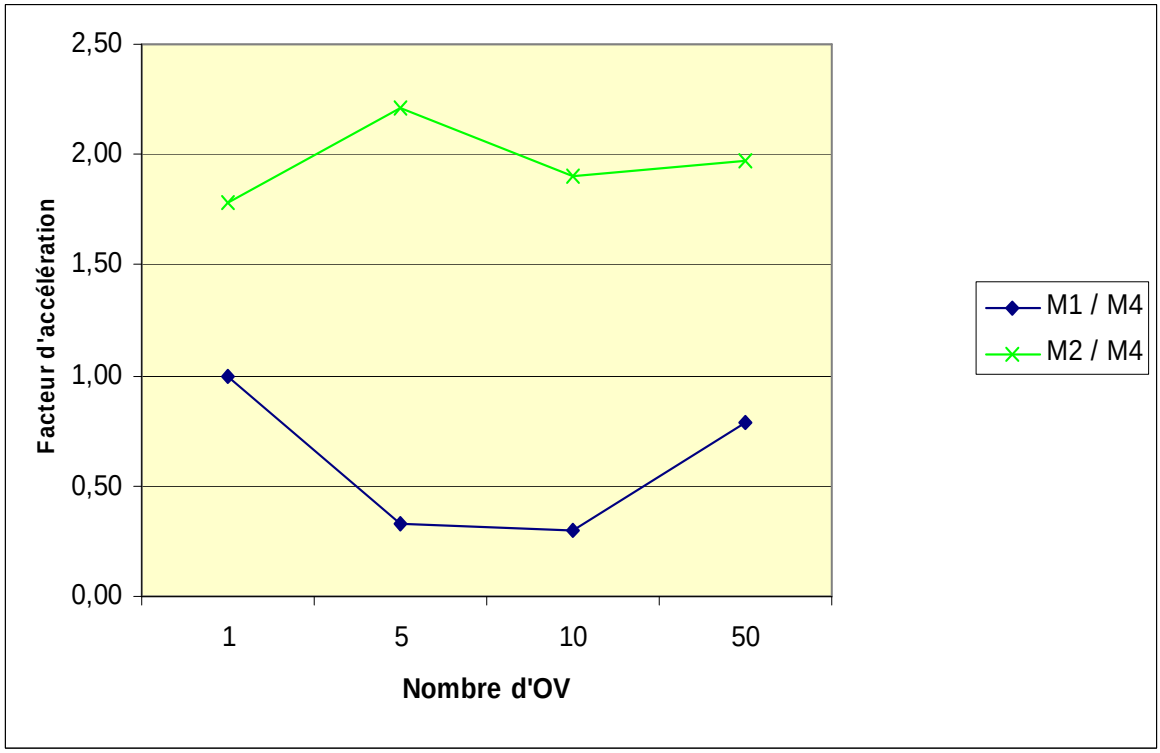


Figure 2 : Facteur d'accélération de M4 par rapport à M1 et M2 pour la découverte de sources de données par rapport au nombre d'OVs

Nous pouvons constater que notre méthode améliore de 5 fois en moyenne le temps de réponse par rapport à M3. Nous obtenons également une amélioration du temps de réponse lorsqu'on augmente le nombre d'OV. Les temps de réponses générés par M4 sont entre 1,7 et 2 fois plus réduits que les temps de réponses générés par la méthode M2. Cependant, les temps de réponses obtenus par la méthode M1 sont meilleurs que les nôtres (entre 29 % et 33 %). L'écart est plus réduit lorsqu'on expérimente avec plus de 10 OVs. Néanmoins, comme nous l'avons fait remarquer précédemment, cette méthode est moins précise concernant la prise en compte de la sémantique. En effet, nous sacrifions la précision et la complétude dans les réponses en utilisant ce type de méthode.

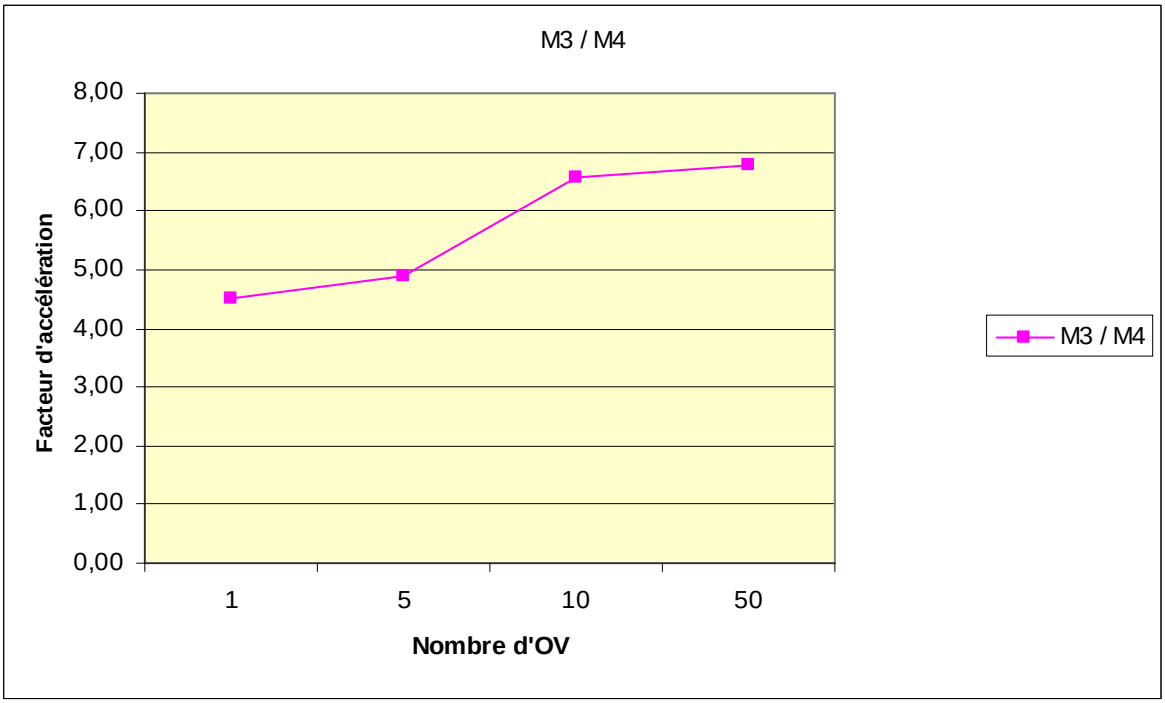


Figure 3 : Facteur d'accélération de M4 par rapport à M3 pour la découverte de sources de données par rapport au nombre d'OVs

En conclusion, nous pouvons constater que notre méthode améliore substantiellement les temps de réponse par rapport aux méthodes M2 et M3. Ces méthodes, comme la nôtre, utilisent des liens de 'mapping' pour traduire un concept d'une ontologie vers une autre. Enfin, notre méthode obtient de moins bonnes performances que M1 mais elle est plus précise sémantiquement. Cette précision est au détriment de temps de réponse supérieur dans notre méthode de découverte de sources de données.

3.2. Évaluation des performances pour plusieurs requêtes de découverte

Nous nous sommes également intéressés à l'impact du nombre de requêtes soumises par seconde sur les temps de réponse. Dans cette perspective, nous avons fait varier le nombre de requêtes soumises par seconde et par OV. Cela équivaut au nombre de requêtes de découverte inter-ontologies de domaines

envoyé par seconde. Les courbes représentées dans les figures 4 et 5 montrent l'impact du nombre de requêtes sur le temps de réponse.

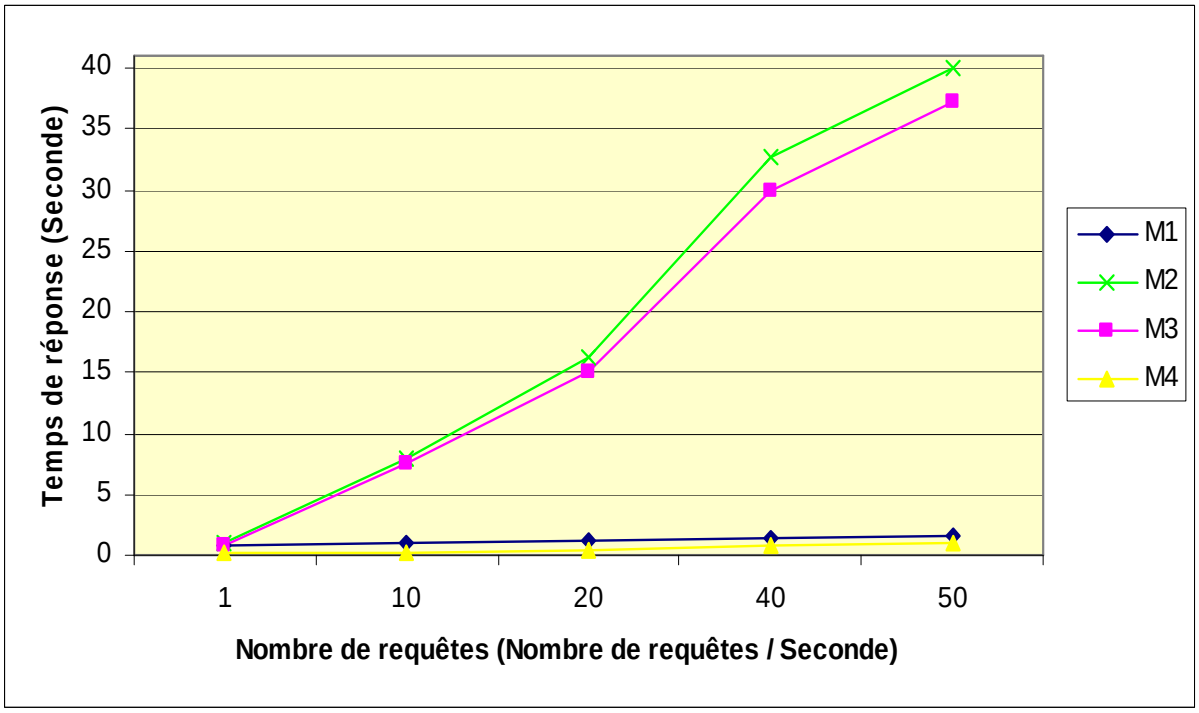


Figure 4 : Impact du nombre de requêtes soumises par seconde sur le temps de réponse avec 5 OVs

Nous avons également expérimenté des systèmes avec 5 et 10 OVs. Nous pouvons constater dans les figure 4 et 5 que lorsque le nombre de messages de découverte par seconde augmentent, les temps de réponse obtenus avec notre méthode sont nettement inférieurs à ceux obtenus avec les méthodes M2 et M3. Dans le cas d'utilisation de 10 OVs et lors de l'envoi de 40 requêtes par seconde, nous constatons un temps de réponse d'environ 1,5 secondes pour notre méthode alors que les méthodes M2 et M3 génèrent des temps de réponse de plus de 58 secondes (voir figure 5). Ceci représente respectivement une accélération d'environ 37 fois. Cette différence accroît encore plus lorsque le nombre de requêtes par seconde augmente. L'amélioration de ces temps de réponses par notre méthode peut atteindre 50 fois par rapport à M2 et M3 en présence de 50 requêtes par seconde dans chacune des 10 OVs utilisés. Ces

temps de réponses élevés pour M2 et M3 sont dus principalement à certains pairs qui forment des goulets d'étranglement.

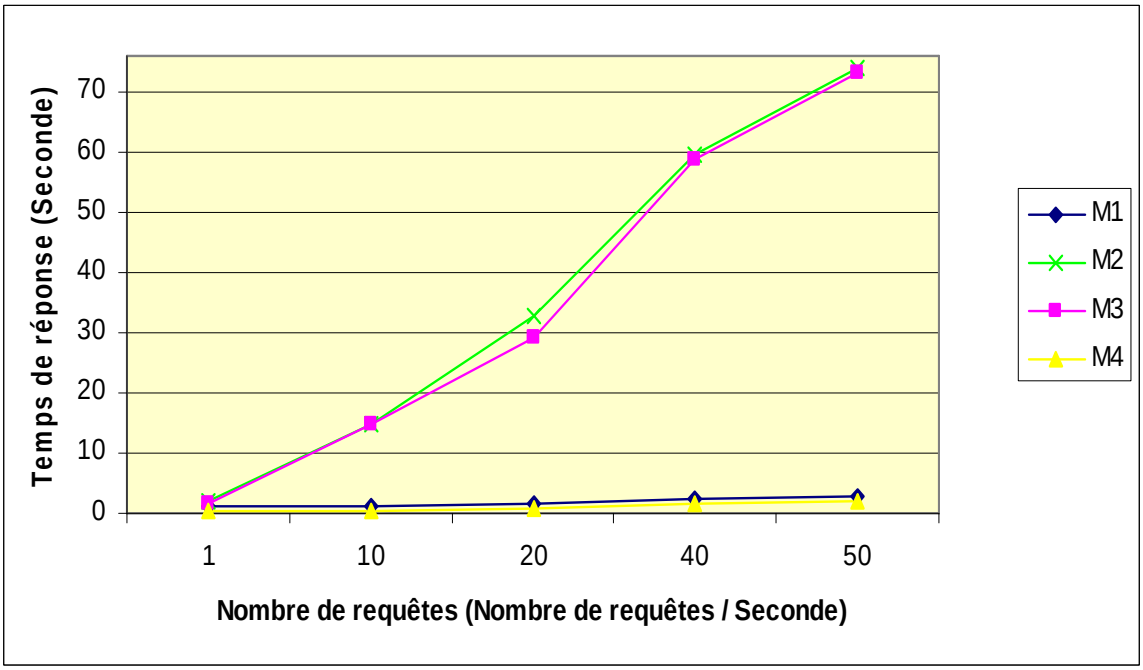


Figure 5 : Impact du nombre de requêtes soumises par seconde sur le temps de réponse avec 10 OVs

Dans les figures 4 et 5, nous observons que notre méthode est légèrement plus rapide que la méthode M1. Cette accélération est encore plus importante lorsque le nombre de messages par seconde est plus important. Cette amélioration est due au fait que la procédure d'envoi dans notre méthode est guidée suivant des liens de correspondances entre les OVs. En effet, dans la méthode M4, la requête de découverte de sources de données n'est pas forcément envoyée à tous les pairs du système comme c'est le cas dans la découverte par la méthode M1 (nombre de requêtes diffusées réduit). De plus, dans notre proposition, il existe plusieurs points de sortie entre deux OVs. En effet, la soumission de plusieurs requêtes au sein d'une OV dans notre méthode fait intervenir différents points de sortie alors que l'utilisation de la diffusion de requêtes dans M1 engendre une saturation du réseau et des goulets d'étranglement sur certains pairs.

Nous nous sommes également intéressés au facteur d'accélération des temps de réponses entre les quatre méthodes comparées lorsque le nombre de requêtes par seconde varie entre 1 et 50. Ceci est fait dans le cas d'utilisation de 5 et 10 OVs. La figure 6 (respectivement 7) montre le facteur d'accélération de M4 par rapport à M1, M2 et M3 pour la découverte de sources de données en variant le nombre de requêtes soumises par seconde pour 5 OVs (respectivement 10 OVs).

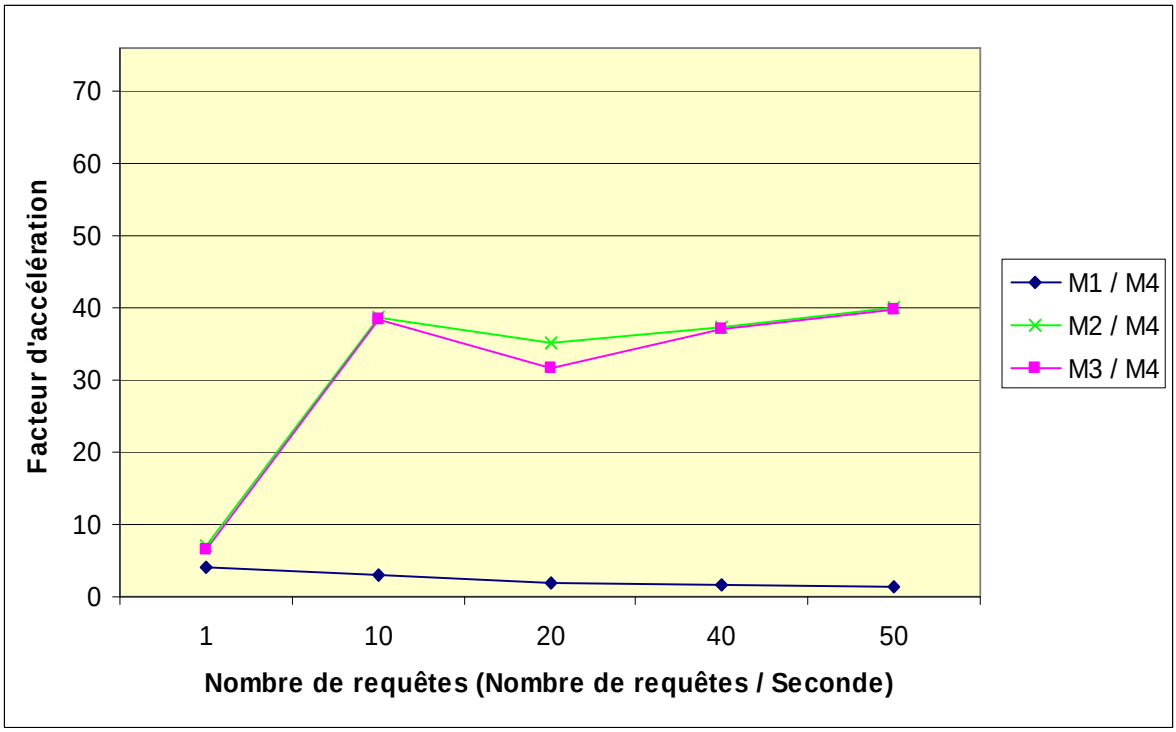


Figure 6 : Facteur d'accélération de M4 par rapport à M1, M2 et M3 pour la découverte de sources de données en variant le nombre de requêtes soumises par seconde (5OVs)

Pour un nombre d'OV fixé à 5, nous pouvons remarquer que les temps de réponses générés par notre méthode sont 40 fois plus réduits en moyenne que les temps de réponses générés par les méthodes M2 et M3 lorsque le nombre de requête par seconde varie entre 10 et 50.

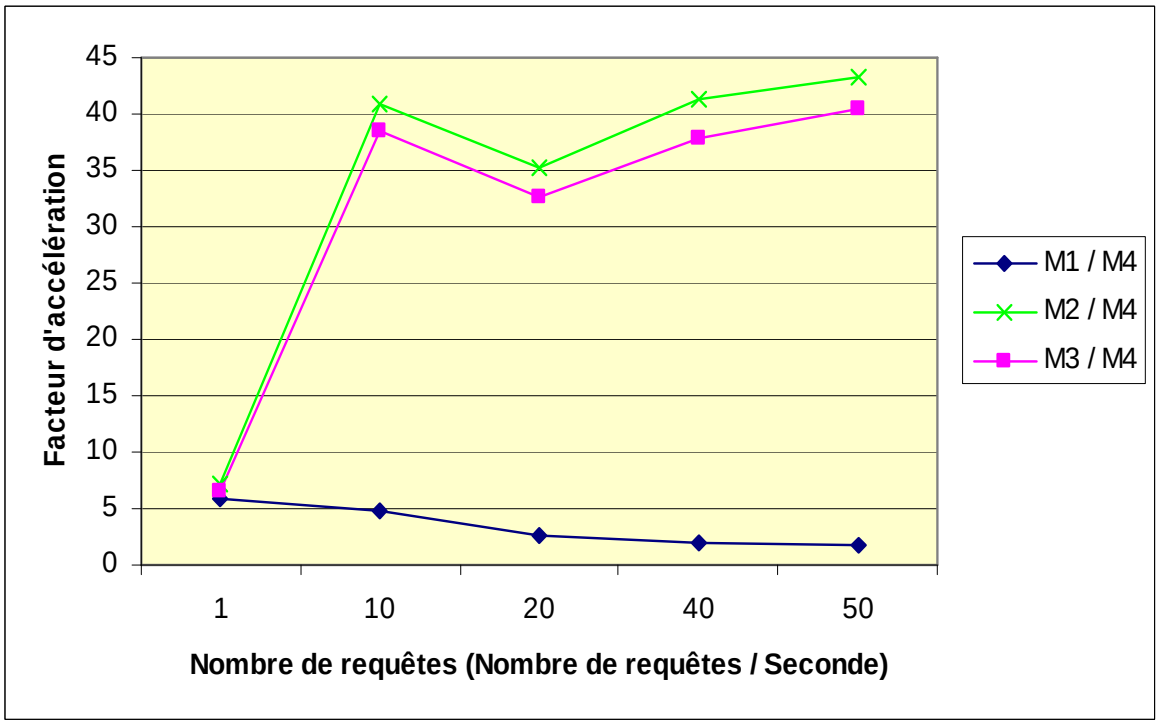


Figure 7 : Facteur d'accélération de M4 par rapport à M1, M2 et M3 pour la découverte de sources de données en variant le nombre de requêtes soumises par seconde (10OVs)

Dans le cas de simulation d'un système avec 10 OVs, nous remarquons que l'écart entre les temps de réponses de notre méthode et les deux méthodes M2 et M3 se creuse. Ainsi, nous constatons une amélioration de 45 fois par rapport à ces deux méthodes M2 et M3. Si nous comparons nos temps de réponses à ceux obtenus par la méthode M1, nous constatons par exemple, que pour 5 OVs et pour un nombre de requêtes soumises par seconde compris entre 10 et 50 l'amélioration est comprise entre 2 et 4 fois. Quand il s'agit de 10 OVs et pour un nombre de requêtes par seconde compris entre 10 et 50 l'amélioration se confirme. Elle est comprise entre 1,50 et 3 fois. Par la suite, nous pouvons déduire une amélioration de 2 fois en moyenne.

En résumé, nous pouvons conclure que notre méthode améliore substantiellement les temps de réponses par rapport aux méthodes M2 et M3. Comparée à la méthode M1 basée sur la diffusion, notre méthode produit de

moins bonnes performances lorsque le nombre de requêtes par seconde est réduit (moins de 5 requêtes par seconde). Néanmoins, notre méthode produit des résultats plus précis du fait qu'elle se base sur les liens de 'mappings' existant entre ontologies de domaine. Cette précision est au détriment de 30 % au max de temps de réponse supérieur par rapport au temps de réponses généré par la méthode M1. Les expériences avec plusieurs requêtes de découverte par seconde ont montré que notre méthode est meilleure. En effet, lorsque le nombre de requêtes est supérieur à 10 requêtes par seconde, nous obtenons de meilleures performances par rapport à toutes les autres méthodes. A titre d'exemple, les temps de réponses sont deux fois inférieures à ceux obtenus par la méthode M1 en plus d'une meilleure prise en charge des aspects sémantiques pour notre méthode.

4. Maintenance du système

Dans cette section, nous évaluons l'impact de la dynamicité des pairs, fortement présente dans les environnements à grande échelle telle que les Grilles, sur le nombre de messages nécessaire à la maintenance du système. Pour cela, nous mettons en évidence l'effet de la connexion / déconnection des pairs sur le nombre de messages nécessaires pour maintenir le système après la connexion / déconnection d'un certain nombre de pairs (variable dans nos courbes). En d'autres termes, nous étudions le nombre de messages nécessaires pour maintenir à jour les différentes structures utilisées dans les quatre méthodes comparées. Une telle maintenance nécessite deux types de mise à jour : (i) la mise à jour de la DHT à chaque connexion / déconnection de pairs et (ii) la mise à jour des points de sortie uniquement lors de la connexion d'un pair. En conséquence, les messages de maintenance représentent les messages de mise à jour de la DHT et les messages nécessaires pour la mise à jours des points de sortie (notre méthode) ou ceux utilisés pour informer les autres pairs du système (cas des autres méthodes). Il est clair que la maintenance de la DHT génère un coût de maintenance important surtout lorsque cette connexion / déconnexion concerne un nombre important de pairs. Néanmoins, ceci est valable pour l'ensemble des quatre méthodes comparées dans cette section.

Cette maintenance est d'autant plus importante qu'elle concerne également les points de sorties. En conséquence, une mise à jour de ces points s'avère nécessaire. Dans ce qui suit, la différence entre les performances des différentes méthodes comparées intervient notamment lors de la mise à jour de ces points de sorties.

Dans cette section, nous avons simulé la connexion et la déconnexion de ces pairs. La connexion d'un pair correspond à l'arrivée de celui-ci dans l'OV de son domaine. Quant à la déconnexion d'un pair, elle peut être suite à une panne ou un départ volontaire du système. Nous avons montré dans les chapitres suivant que la connexion / déconnexion de pairs génère des coûts de maintenance élevés surtout lorsqu'il s'agit de connexions / déconnexions fréquentes dans le système. Pour mesurer la validité de la méthode proposée, nous varions le nombre de pairs qui se connectent et / ou se déconnectent du système. Puis, nous comparons le nombre de messages nécessaire à la mise à jour d'un tel système dans les quatre méthodes comparées.

La figure 8 (respectivement la figure 9) montre le nombre de messages nécessaire à la mise à jour du système lorsqu'un nombre de pairs que nous faisons varier arrive ou quitte le système composé de 5 OVs (respectivement 10 OVs).

Plus le nombre de pairs qui se connectent ou se déconnectent au système est important plus le nombre de messages de mise à jour est important. La méthode M2 génère le plus important coût de maintenance. En effet, si un pair super pair quitte ou arrive dans le système, tous les pairs dits 'feuilles' doivent être mis à jour. La méthode M3 génère également d'importants coûts de maintenance vu la topologie utilisée. Nous pouvons noter que plus le nombre de pairs qui se connectent ou se déconnectent augmente plus l'écart entre les courbes des quatre méthodes s'élargit. Dans les méthodes super-pair (M2) et deux à deux (M3), le nombre de messages nécessaires pour maintenir le système après la connexion ou déconnexion des pairs est de plus de 14000

messages à partir de 10 pairs connectés ou déconnectés. Lorsque le nombre d'OVs est de 10, ce nombre dépasse 20000 messages. Cependant, la méthode par inondation (M1) exige un nombre moins élevé. Dans les figures 8 et 9 ce nombre de messages est de 12000 messages (respectivement 18000 messages) lors de la connexion ou déconnexion de 10 pairs aux 5 OVs (respectivement 10 OVs).

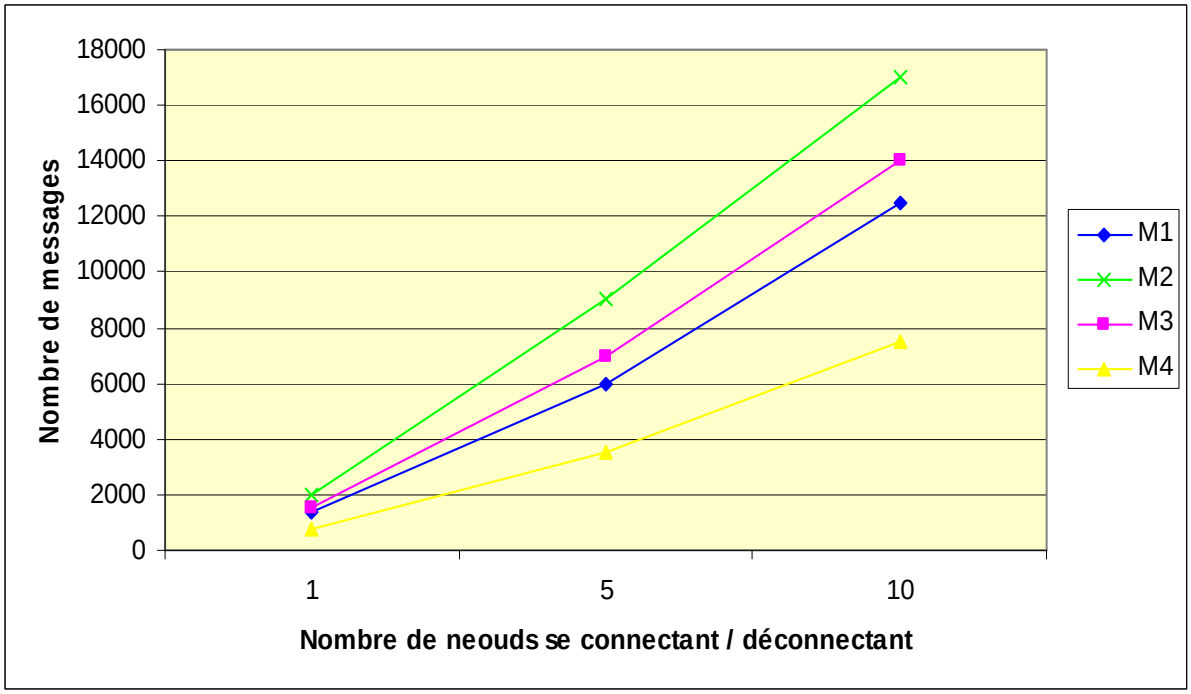


Figure 8 : Impact du nombre de pairs se connectant / déconnectant sur le nombre de messages nécessaires à la maintenance du système contenant 5 OVs

Notre méthode nécessite moins de messages pour la mise à jour du système que toutes les autres méthodes comparées. Ainsi, seulement 7200 messages (respectivement 11000 messages) sont nécessaires à la maintenance du système lorsque 10 pairs arrivent ou quittent le système dans une configuration de 5 OVs (respectivement 10 OVs). L'emploi de la maintenance paresseuse dans notre système permet un gain important au niveau de ce nombre de messages du fait que le nombre de messages nécessaires pour mettre à jour les points de sortie est beaucoup moins important avec notre méthode. Ainsi, lors de la connexion d'un pair, les points de sorties sont établis

lors de la découverte de sources de données. Les différents pairs qui référencent par des points de sorties qui se déconnectent ne sont pas également informés lors de la déconnexion de points de sorties. La mise à jour est faite lors du processus de découverte.

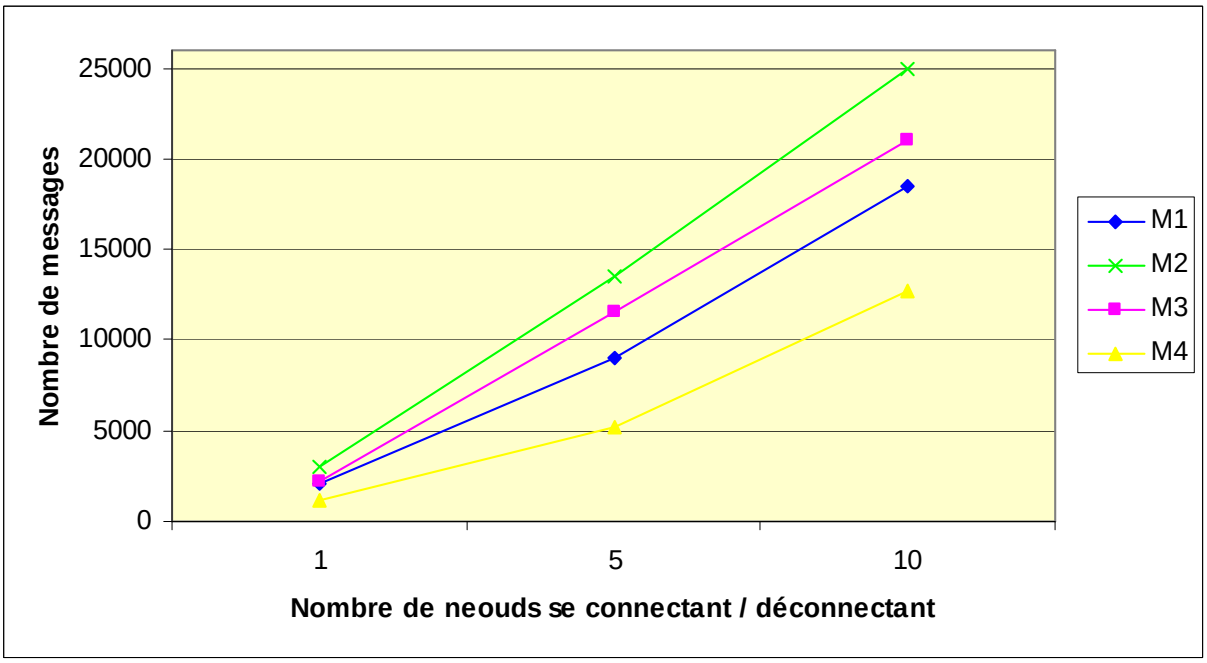


Figure 9 : Impact du nombre de pairs se connectant / déconnectant sur le nombre de messages nécessaires à la maintenance du système contenant 10 OVs

Nous remarquons également que plus on augmente le nombre d'OVs utilisées, plus l'écart en nombre de messages se creuse entre notre méthode et les autres méthodes comparées. En effet, le nombre de messages nécessaires à la mise à jour est moindre par rapport à la méthode M1 dans ce cas. Ceci montre que notre méthode est plus efficace quand il s'agit de connexion ou déconnexions fréquentes ce qui est très probable dans des environnements à grande échelle.

5. Conclusion

L'évaluation des performances nous a permis de comparer notre méthode de découverte de sources de données proposée (M4) aux trois autres méthodes (M1, M2, M3). Toutes ces méthodes utilisent une DHT pour la découverte de sources de données intra-ontologie de domaine. Cela rend leurs résultats équivalents pour ce type de découverte. Pour la découverte de sources de données inter-ontologies de domaines, nous avons constaté que les deux méthodes ; M2 et M3 auxquelles nous avons comparé notre méthode imposent le respect d'une certaine topologie de 'mapping'. L'évaluation de performances de notre méthode montre une réduction significative des temps de réponses par rapport aux méthodes M2 et M3. Le gain est d'autant plus important lorsqu'il s'agit d'expériences avec plusieurs requêtes de découverte de sources de données par seconde. Ce gain augmente également lorsque les systèmes comportent un plus grand nombre d'OVs, ce qui est souvent le cas dans les environnements à grande échelle. Comparée à la méthode basée sur la diffusion (M1), notre méthode présente des temps de réponses légèrement supérieurs dans le cas d'une seule requête de découverte. Cependant, la méthode M1 suit un processus de découverte plutôt aléatoire adoptant la recherche par inondation et donc la recherche par mot clés, ce qui risque de générer des résultats moins précis et même erronés. Les résultats de la découverte à travers la méthode M4 sont alors plus précis d'autant plus que notre méthode s'adapte à n'importe quelle topologie de 'mapping' entre les ontologies et que les concepts sont traduits d'une ontologie de domaine vers une autre. Sacrifier alors des temps de réponses plus élevés (de 30 %) pour avoir en contre partie un résultat sûr et avec un maximum de complétude, semble être meilleur. Néanmoins, lorsque le nombre de requêtes soumises par seconde augmente, notre méthode présente de meilleures performances par rapport à la méthode M1. Dans un environnement à grande échelle, il semble plus raisonnable d'avoir plus de 10 requêtes par seconde que d'avoir une seule requête par seconde. Ainsi, en plus de tenir compte de la sémantique, nous pensons que notre méthode est mieux armée pour passer à l'échelle.

Quant à la maintenance du système, l'évaluation de notre méthode montre de très bons résultats. En effet, grâce à la maintenance paresseuse, les coûts de maintenance du système sont réduits d'une façon significative par rapport à l'ensemble des trois autres méthodes auxquelles nous avons comparé nos performances.

Chapitre 5 : Conclusion et perspectives

Chapitre 5 : Conclusion et perspectives

Dans cette thèse, nous avons traité le problème de la découverte de ressources dans un environnement à grande échelle en prenant en compte l'hétérogénéité sémantique des données et la dynamicité de pairs. Les ressources à découvrir peuvent être classés en deux types : les ressources de calcul (e.g, processeur Intel avec une mémoire vive d'au moins 2Go…) et les ressources de type source de données (e.g, un fichier, une base de données…). Nous avons montré l'importance de la découverte de sources de données puisque leur absence entraine l'impossibilité d'évaluation de la requête utilisateur. Dans cette thèse, nous nous sommes uniquement intéressés à la découverte de ressources de type sources de données. Nous avons d'abord présenté une synthèse concernant les différentes approches existantes de découverte de sources de données. Nous avons introduit les approches traditionnelles de découverte de sources de données basés sur les mots clés. Les premières solutions basées sur des systèmes centralisés ne permettent pas le passage à l'échelle puisqu'elles conduisent à un goulet d'étranglement au niveau du serveur central limitant ainsi les performances de la découverte. Beaucoup de travaux de recherche ont essayé de décentraliser le contrôle en s'appuyant sur les systèmes pair-à-pair.

Dans les systèmes pair-à-pair non-structurés, la maintenance s'effectue à faible coûts grâce à un haut degré d'autonomie entre les pairs. Par contre, elles conduisent, très rapidement, à une saturation de réseau à cause du mécanisme de diffusion utilisé dans ces systèmes lors du processus de découverte de sources de données.

Dans les systèmes pair-à-pair structurés, les différentes méthodes de découverte de ressources proposées ont montré une meilleure fiabilité avec de bonnes propriétés de passage à l'échelle. En contrepartie, elles nécessitent un coût de maintenance élevé dû à la dynamicité des pairs.

Quant aux systèmes hybrides, ils combinent les avantages et les inconvénients des systèmes structuré et non-structuré. Cependant, toutes ces méthodes ne prennent pas en compte les problèmes liés à l'hétérogénéité. Plusieurs travaux de recherche se sont intéressés à la prise en compte de ces problèmes dans la découverte de source de données. Mais, ce sont plutôt les problèmes d'hétérogénéité structurelle (e.g., différence de format, structure, complexité des schémas de données) qui ont été traités dans la littérature. Dans ce contexte, nous nous sommes focalisés sur la découverte de sources de données prenant en compte les problèmes d'hétérogénéité sémantique (e.g., synonymie, polysémie) dans un environnement à grande échelle et dynamique.

Nous nous sommes intéressés aux travaux prenant en compte cet aspect dans la découverte de ressources. Les premières approches étaient basées sur les correspondances entre les noms utilisés dans les schémas des sources de données. Cependant, la maintenance des correspondances dans un environnement fortement dynamique constitue une difficulté majeure dans ce contexte.

L'utilisation d'une ontologie globale ne constitue pas également une bonne solution. En effet, concevoir une telle ontologie demeure une tâche complexe vu le nombre gigantesque et grandissant de sources de données dans des domaines tel que le domaine biomédical.

Plusieurs travaux ont alors proposé l'usage de plusieurs ontologies de domaine. Dans cette approche, qui s'est avérée plus prometteuse, une ontologie de domaine est associée à chaque domaine. Des liens de correspondance dits 'mappings' entre ces ontologies sont alors définis. Cependant, toutes les méthodes de découverte de ressources proposées dans ce contexte imposent une topologie pour les relations de 'mapping' ce qui constitue un inconvénient majeur. En effet, la définition de relations de 'mapping' entre ontologies de domaine est une tâche ardue. Or, il existe sur internet des ontologies de domaine ainsi que des relations de 'mapping' entre ces ontologies. La

topologie entre ces relations de ‘mapping’ est quelconque. Un bon défi consiste à réutiliser ces relations de ‘mappings’ entre ontologies sans imposer une topologie fixe entre elles. De plus, une méthode de découverte de sources de données doit être efficace non seulement par rapport à la recherche qui doit prendre en compte l’aspect sémantique mais aussi par rapport à la maintenance à cause de la dynamicité des sources de données.

A partir de ces constatations, nous avons proposé une méthode de découverte de sources de données dans un environnement à grande échelle tout en prenant en compte l’hétérogénéité sémantique des sources de données et leur dynamicité. Dans notre méthode, nous nous basons sur les relations de ‘mappings’ déjà existantes. Nous associons également une ontologie de domaine à chaque domaine. Nous n’imposons aucune topologie pour les relations de ‘mappings’ entre ces ontologies. Une organisation virtuelle (OV) est associée à chaque domaine. Au sein d’un même domaine, les pairs sont connectés suivant une Table de Hachage Distribuée (THD). Ensuite, nous avons proposé un système d’adressage permettant un accès permanent de n’importe quelle ontologie vers une autre. Notre unique hypothèse est que le graphe de ‘mappings’ entre ces ontologies soit connexe. Parmi les pairs d’une OV_i, nous définissons des pairs dits ‘points de sorties’ référençant d’autres pairs dans OV_j de telle sorte qu’un pair appartenant à OV_j ne soit pas référencé par plusieurs points de sorties appartenant à OV_i. Ceci permet d’éviter des goulets d’étranglement au niveau de OV_i. Cela évite également d’avoir des pairs constituant un unique point d’échec (‘single point of failure’). La diffusion des requêtes de découverte de sources de données au sein d’une OV se fait suivant le routage classique utilisé dans une THD. On parle alors de requêtes intra-ontologie. Concernant les requêtes inter-ontologies, on se base sur le système d’adressage mis au point à travers les points de sortie. Dans ce contexte et afin de limiter le nombre des messages nécessaires à la mise à jour de ces points de sorties, une mise à jour dite « paresseuse » est adoptée lors de la connexion ou déconnexion de pairs. En effet, la mise à jour n’est pas faite au moment de la déconnection d’un pair mais lors du processus de découverte de

sources de données. Cela permet une réduction significative des coûts de maintenance surtout en présence d'un effet 'Churn'.

Afin de valider notre méthode, nous l'avons évalué par simulation. Nous avons comparé les performances de notre méthode à celles obtenues par trois autres méthodes de découverte de ressources prenant en compte l'aspect sémantique. Par rapport aux deux méthodes, utilisant comme nous les liens de 'maping', les mesures de performance ont montré une amélioration significative du temps de découverte de sources de données notamment en ce qui concerne les requêtes inter-ontologies de domaines. Nous avons également comparé notre méthode à une autre méthode basée sur le principe de la diffusion. Pour un nombre réduit de requêtes par seconde, nous avons constaté des temps de réponse légèrement plus important pour notre méthode. Dans notre méthode, nous sacrifions un peu de temps de réponse pour avoir en contre partie un résultat plus précis et complet. En effet, la recherche des correspondances de la méthode de diffusion en utilisant des mots clés peut amener à des résultats incomplets voir erronés. Dans notre méthode les concepts sont traduits suivant des règles de 'mapping' d'une ontologie à une autre, ce qui aboutit à un résultat plus précis et complet. Néanmoins, plus le nombre de requêtes par seconde augmente (à partir de 10 requêtes par seconde) plus les temps de réponses de notre méthode deviennent meilleurs par rapport à la méthode de diffusion. Ceci permet de constater une meilleure disposition de passage à l'échelle pour notre méthode tout en prenant en compte de l'aspect sémantique. Les évaluations de performances montrent également une réduction significative des coûts de maintenance du système notamment lors de l'arrivée et du départ d'un nombre important de pairs. Cette réduction est due à une maintenance paresseuse de notre système d'adressage. Ce pendant, le mécanisme est efficace uniquement si le processus de découverte de sources de données est utilisé de manière fréquente. Notre méthode semble particulièrement adaptée à un environnement à grande échelle et dynamique constitué de plusieurs domaines et comportant un grand nombre de sources de

données hétérogènes. Un bon exemple d'application serait la grille de données biomédicale.

L'évaluation de performances semble montrer de bonnes perspectives pour notre méthode. Afin de valider et vérifier qu'elle passe à l'échelle, nous espérons évaluer notre méthode de découverte de ressources en utilisant une plateforme d'expérimentation comportant un nombre important de sources de données fortement hétérogènes. Enfin, le nombre important de sources de données et la dynamicité des pairs posent également un problème majeur lors de l'allocation de ressources pour un plan d'exécution associé à une requête. A notre connaissance, il n'existe pas de méthode d'allocation fiable à grande échelle permettant de prendre en compte la dynamicité des pairs. Nous envisageons de concevoir une méthode d'allocation dynamique de requêtes s'adaptant à l'évolution des ressources.

Bibliographie

Bibliographie

[Alk08] R. Alking, A. Hameurlain and F. Morvan: Ontology-Based Data Source Localization in a Structured Peer-to-Peer Environment. IDEAS, Coimbra, Portugal, 2008

[Alk10] R. Alking: Localisation de sources de données et optimisation de requêtes réparties en environnement pair-à-pair. Thèse de Doctorat, Université Toulouse III Paul Sabatier, 2010

[Are03] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller and J. Mylopoulos: The Hyperion Project: From Data Integration to Data Coordination. SIGMOD Record Journal, 2003

[Ban04] F. Banaei-Kashani, C.-C. Chen, and C. Shahabi: WSPDS: Web Services Peer-to-peer Discovery Service. International Conference on Internet Computing, Las Vegas, Nevada, USA, 2004

[Bos91] Robert Bosch GmbH. CAN Specification Version 2.0, 1991

[Cas04] S. Castano, A. Ferrara, S. Montanelli, G. Racca: Semantic Information Interoperability in Open Networked Systems. 2004

[Cha03] Y. Chawathe, S. Ratnasamy, L. Breslau: Making Gnutella-like P2P Systems. Scalab, 2003.

[Cla00] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong.: Freenet: A distributed anonymous information storage and retrieval system. Workshop on Design Issues in Anonymity and Unobservability, 311-320, ICSI, Berkeley, CA, USA, July 2000

[Com09] C. Comito, C. Mastroianni and D. Talia: A Semantic-aware Information System for Multi-Domain Applications over Service Grids. IPDPS, Rome, Italy, 2009

[Cre02] A. Crespo and H. Garcia-Molina, \Routing Indices for Peer-to-Peer Systems". Proc. 22nd Int. Conf. on Distributed Computing Systems (ICDCS'02), pp. 23-30, 2002

[Cre03] A. Crespo and H. Garcia-Molina, \Semantic Overlay Networks for P2P Systems". Technical Report, Stanford University, 2003

[Cru04a] I. F. Cruz, H. Xiao, F. Hsu: An Ontology-based Framework for XML Semantic Integration. 2004

[Cru04b] I. F Cruz., H. Xiao and F. Hsu: Peer-to-Peer Semantic Integration of XML and RDF Data Sources. International Workshop on Agents and Peer-to-Peer Computing AP2PC, New York, NY, USA, 2004

[Fis] Fisk, A. A.: Gnutella Dynamic Query Protocol v0.1., http://www.thegdf.org/wiki/index.php?title=Dynamic Querying

[Elm05] E. Elmroth and J. Tordsson: An interoperable, Standards based Grid Resource Broker and Job Submission Service. The International Conference on e-Science and Grid Computing, Melbourne, Australia, 2005

[Fay06] D. Faye, G. Nachouki and P. Valduriez, SenPeer : Un système Pair-à-Pair de médiation de données. ARIMA Journal, 2006

[Fay07] David Faye, Gilles Nachouki, and Patrick Valduriez, Semantic Query Routing in SenPeer, a P2P Data Management System. LNCS 4658, pp. 365-374, 2007

[Fil04] Filho et al.; PerDiS: a scalable resource discovery service for the ISAM pervasive environment, International Workshop on Hot Topics in Peer-to-Peer Systems, pp. 80-85, Oct. 2004

[Fip95] FIPS 180-1: Secure Hash Standard. National Technical Information Service, U.S. Department of Commerce/NIST Springfield, VA, April 1995

[Gia05] P. Rodríguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. J. Miller, J. Mylopoulos: Data Sharing in the Hyperion Peer Database System. Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

[Gka05] C. Gkantsidis, M. Mihail and A. Saberi, \Hybrid Search Schemes for Unstructured Peer-to-Peer Networks". Proc. IEEE INFOCOM, 2005

[Gnu00] Gnutella Development Home Page. http://gnutella.wego.com/, 2000

[Haa04] P. Haase, J. Broekstra, M. Ehrig , M. Menken, P. Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, C. Tempich: Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. Proceedings of the International Semantic Web Conference (ISWC), Hiroshima, Japan, November 2004

[Hal03] Alon Y. Halevy, Zachary G. Ives, Peter Mork and Igor Tatarinov: Piazza: Data Management Infrastructure for Semantic Web Applications. WWW, Budapest, Hungary, 2003

[Ham09] D. Cokuslu, A. Hameurlain, K. Erciyes: Grid Resource Discovery Based on Web Services. ICITST, 2009

[Har03] N. Harvey and al.: Skipnet: A Scalable Overlay Network with Practical Locality Proprieties. Proc of USITIS, Seattle, USA, 2003

[Hei04] A. Heine, M. Hovestadt and O. Kao: Towards Ontology-Driven P2P Grid Resource Discovery Felix. IEEE/ACM International Workshop on Grid Computing (GRID), Pittsburgh, Pennsylvania, USA, 2004

[Iam01] A. Iamnitchi: On Fully Decentralized Resource Discovery in Grid Environments. Lecture Notes in Computer Science, Book Grid Computing, Springer, Berlin, Heidelberg, Vol 2242, 51-62, 2001

[Jag05] H. V. Jagadish, B. C. Ooi, Q. H. Vu: BATON: A Balanced Tree Structure for Peer-to-Peer Networks. VLDB, 661-672, 2005

[Jon08] C. Jonquet, M. A. Musen and N. Shah: A System for Ontology-Based Annotation of Biomedical Data. DILS, Evry, France, LNBI 5109, Springer-Verlag Berlin Heidelberg: 144-152, 2008

[Kal02] V. Kalogeraki, D. Gunopoulos and D. Z. Yazti, “A local search mechanism for peer-to-peer networks”. ACM Int. conf. on Information and Knowledge Management (CIKM’02), pp. 300-307, 2002

[Kau07] D. Kaur and J. Sengupta.: Resource discovery in web services based grids. World Academy Of Science Engineering And Technology, 2007

[Kaz01] Kazaa: Homepage, http:// www.Kazaa.com, 2001

[Kem03] A. Kementsietsidis, M. Arenas and R. Miller: Managing Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. SIGMOD, San Diego, California, USA, 2003

[Ket10] Imen Ketata, Riad Mokadem, Franck Morvan, Abdelkader Hameurlain, “Efficient Semi-Automatic Maintenance of Mapping Between Ontologies in a Biomedical Environment ”, dans : International Conference on Enterprise Information Systems (ICEIS 2010), Funchal, Madeira - Portugal, 08/06/2010-12/06/2010, Vol. 1, r (Eds.), INSTICC Press, (support électronique), Juin 2010

[Ket11a] Imen Ketata, Riad Mokadem, Franck Morvan, “Biomedical Resource Discovery considering Semantic Heterogeneity in Data Grid Environments”, dans: International Conference on Integrated Computing Technology (InTech 2011), Sao Carlos-Brazil, Mai-Juin 2011

[Ket11b] Imen Ketata, Riad Mokadem, Franck Morvan, “Resource Discovery Considering Semantic Properties in Data Grid Environments” (regular paper), dans : International Conference on Data Management in Grid and P2P Systems (GLOBE 2011), Toulouse, 01/09/2011-02/09/2011, Springer, LNCS 6864, p. 61-72, Septembre 2011

[Koo06] S. G. M. Koo, K. Kannan, C.S. George Lee: On neighbour selection strategy in hybrid Peer-to-Peer networks. Future Generation Computer Systems, Vol. 22, 732-741, 2006

[Li05] J. Li and S. Vuong. Semantic Overlay Network for Grid Resource Discovery. Grid Computing Workshop, Seattle, Washington, USA, 2005

[Lia06] J. Liang, R. Kumar, K. W. Ross: The FastTrack overlay: A measurement study. Computer Networks (50), 842-858, 2006

[Mas05a] C. Mastroianni, D. Talia, O. Verta: A super-peer model for resource discovery services in large-scale Grids. Future Generation Computer Systems, Elsevier Science, Vol. 21, 1235-1248, 2005

[Mas05b] C. Mastroianni, D. Talia and O. Verta, \A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis". Proc. European Grid Conference (EGC 2005), LNCS, vol. 3470, pp. 132-143, Springer 2005

[May02] P. Maymounkov, D. Mazieres, Kademlia: A Peer-to-Peer information system based on the XOR metric. Proc. First Int. Workshop on Peer-to-Peer Systems (IPTPS), 53-65, 2002

[Mes08] E. Meshkova et al.: A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. International Journal of Computer and Telecommunications Networking Archive, Vol. 52 (11), 2097-2128, 2008

[Mol08] G. Moltó, V. Hernandez and J. M. Alonso: A service oriented WSRF-based architecture for metascheduling on computational grids. International Journal of Grid Computing and eScience: Future Generation Computing Systems, 2008

[Nap01] http://www.napster.com, http://music.napster.com/napsterhomemain.htm?signin=1, 2001

[Nav05] I. Navas, I. Sanz, J.F. Aldana, and R. Berlanga: Automatic Generation of Semantic Fields for Resource Discovery in the Semantic Web. DEXA, Bilbao, Spain, 2005

[Ope04] Open Chord home page : http://www.uni-bamberg.de/en/pi/bereich/research/software_projects/openchord/, 2004

[Pac07] E. Pacitti, P. Valduriez, M. Mattoso: Grid Data Management: Open Problems and New Issues. Springer Science Business Media B.V. 2007

[Pap05] C. Papadakis, P. Fragopoulou, E. Athanasopoulos, M. Dikaiakos, A. Labrinidis, E. Markatos: A feedback-based approach to reduce duplicate messages in unstructured Peer-to-Peer networks. Integrated Workshop on Grid Research, 2005

[Pir08a] G. Pirrò, M. Ruffolo and D. Talia: An Algorithm for Discovering Ontology Mappingbs in P2P Systems. KES, Zagreb, Croatia, 2008

[Pir08b] G. Pirrò, D. Talia and P. Trunfio: ERGOT: Combining DHTs and SONs for Semantic-Based Service Discovery on the Grid. CoreGRID, Technical Report, 2008

[Pup05] D. Puppin et al.: A grid information service based on peer-to-peer. EuroPar, Vol 3648, Springer LNCS, 454-464, 2005

[Ram06] T. G. Ramos and A. C. Magalhaes: An extensible resource discovery mechanism for grid computing environments. IEEE International Symposium on Cluster Computing and the Grid (CCGRID), IEEE Computer Society, Singapore, 2006

[Ran08] R. Ranjan, A. Harwood, R. Buyya: Peer-to-peer-based resource discovery In global grids: A tutorial. IEEE Communications Surveys and Tutorials 10(1-4), 6-33, 2008

[Rat01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker: A scalable content-addressable network. Proc. ACM SIGCOMM'01, San Diego, California, USA, Aug. 2001

[Rey03] P. Reynolds and A. Vahdat: Efficient Peer-to-Peer Keyword Searching. Proceedings of the ACMIFIP/USENIX, Middleware, 2003

[Row01] A.Rowstron and P. Druschel Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, November 2001

[Sam09] M. Al Samad: Découverte de ressources pour l'évaluation efficace de requête réparties en environnement de grille de données. Thèse de Doctorat, Université Toulouse III Paul Sabatier, 2009

[Sar03] S. Saroiu, K. P. Gummadi and S. D. Gribble: Measuring and analysing the characteristics of Napster and Gnutella hosts. Multimedia Systems. Vol. 9 (2), 170-184, 2003

[Sar04] C. Sartiani, P.Manghi, G. Ghelli, and G. Conforti: XPeer: A Self-organizing XML P2P Database System. EDBT Workshop on P2P and Databases, Heraklion, Crete, Greece, 2004

[Sch02] M. T. Schlosser, M. Sintek, S. Decker, W. Nejdl: HyperCuP - Hypercubes, Ontologies and Efficient Search on Peer-to-Peer Networks. AP2PC, 112-124, 2002

[Sen02] S. Sen and J. Wang: Analyzing peer-to-peer traffic across large networks, 2002

[Sio05] W. Siong Ng, Beng Chin Ooi, Kian-Lee Tan and Aoying Zhou: PeerDB: A P2P-based System for Distributed Data Sharing. International Conference of Data Engineering, Bangalore, India, 2005

[Sto01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnany: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. SIGCOMM, San Diego, California, USA, 2001

[Tal05] D. Talia and P. Trun¯o, \Peer-to-Peer Protocols and Grid Services for Resource Discovery on Grids". In: L. Grandinetti (Ed.), Grid Computing: The New Frontier of High Performance Computing, Advances in Parallel Computing, vol. 14, Elsevier Science, 2005

[Tru07] P. Trunfio et al.: Peer-to-Peer resource discovery in Grids: Models and systems, Future Generation Computer Systems, Elsevier, Vol. 23 (7), 864-878, 2007

[Yan03] B. Yang, H. Garcia-Molina: Designing a super-peer network. Proc. of the 19th Intl. Conf. on Data Engineering, ICDE, 49-60, 2003

[Yu03] J. Yu, S. Venugopal, and R. Buyya: Grid market directory: A web services based grid service publication directory. Grid Computing and Distributed Systems (GRIDS), Technical report, 2003

[Zei05] D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos: Exploiting locality for scalable information retrieval in Peer-to-Peer systems. Information Systems Journal 30 (4) 277-298, 2005

[Zha01] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, Berkeley, CA, USA, April 2001

[Zhu04a] C. Zhu, Z. Liu, W. Zhang, W. Xiao and J. Huang: An Efficient Decentralized Grid Service Discovery Approach based on Service Ontology. IEEE/WIC/ACM International Conference on Web Intelligence (WI), Beijing, China, pp. 570–573, 2004

[Zhu04b] H. Zhuge, J. Liu, L. Feng, and C. He: Semantic-Based Query Routing and Data Integration in Peer-to-Peer Semantic Link Networks. Computational Intelligence, 2004

Publications

Méthode de découverte de sources de données en environnement de grille de données tenant compte de la sémantique

Imen KETATA

Publications

Conférences internationales

[1] Imen Ketata, Riad Mokadem, Franck Morvan, Abdelkader Hameurlain, “Efficient Semi-Automatic Maintenance of Mapping Between Ontologies in a Biomedical Environment ”, dans : International Conference on Enterprise Information Systems (ICEIS 2010), Funchal, Madeira - Portugal, 08/06/2010-12/06/2010, Vol. 1, r (Eds.), INSTICC Press, (support électronique), Juin 2010

[2] Imen Ketata, Riad Mokadem, Franck Morvan, “Biomedical Resource Discovery considering Semantic Heterogeneity in Data Grid Environments”, dans: International Conference on Integrated Computing Technology (InTech 2011), Sao Carlos-Brazil, Mai-Juin 2011

[3] Imen Ketata, Riad Mokadem, Franck Morvan, “Resource Discovery Considering Semantic Properties in Data Grid Environments” (regular paper), dans : International Conference on Data Management in Grid and P2P Systems (GLOBE 2011), Toulouse, 01/09/2011-02/09/2011, Springer, LNCS 6864, p. 61-72, Septembre 2011

Rapports de recherche

[1] Imen Ketata, Riad Mokadem, Franck Morvan, Abdelkader Hameurlain, “Maintenance semi-automatique de correspondances entre ontologies de domaines dans l’environnement biomédical”, Rapport de recherche IRIT/RR--2009-30—FR, 2009

[2] Imen Ketata, Riad Mokadem, Franck Morvan, “Biomedical Resource Discovery considering Semantic Heterogeneity in Data Grid Environments”, Rapport de recherche IRIT/RR--2011-1—FR, 2011

Méthode de découverte de sources de données tenant compte de la sémantique en environnement de grille de données

Imen KETATA

Résumé

Les applications grilles de données de nos jours partagent un nombre gigantesque de sources de données en un environnement instable où une source de données peut à tout moment joindre ou quitter le système. Ces sources de données sont hétérogènes, autonomes et distribuées à grande échelle. Dans cet environnement, la découverte efficace des sources de données pertinentes pour l'exécution de requêtes est un défi. Les premiers travaux sur la découverte de sources de données se sont basés sur une recherche par mots clés. Ces solutions ne sont pas satisfaisantes puisqu'elles ne tiennent pas compte des problèmes de l'hétérogénéité sémantique des sources de données. Ainsi, d'autres solutions proposent un schéma global ou une ontologie globale. Cependant, la conception d'un tel schéma ou d'une telle ontologie est une tâche complexe à cause du nombre de sources de données. D'autres solutions optent pour l'usage de correspondances entre les schémas des sources de données ou en s'appuyant sur des ontologies de domaine et en établissant des relations de 'mapping' entre ces dernières. Toutes ces solutions imposent une topologie fixe soit pour les correspondances soit pour les relations de 'mapping'. Cependant, la définition de relations de 'mapping' entre ontologies de domaine est une tâche ardue et imposer une topologie fixe est un inconvénient

majeur. Dans cette perspective, nous proposons dans cette thèse une méthode de découverte de sources de données prenant en compte les problèmes liés à l'hétérogénéité sémantique en environnement instable et à grande échelle. Pour cela, nous associons une Organisation Virtuelle (OV) et une ontologie de domaine à chaque domaine et nous nous basons sur les relations de 'mappings' existantes entre ces ontologies. Nous n'imposons aucune hypothèse sur la topologie des relations de 'mapping' mis à part que le graphe qu'elles forment soit connexe. Nous définissons un système d'adressage permettant un accès permanent de n'importe quelle OV vers une autre malgré la dynamicité des pairs. Nous présentons également une méthode de maintenance dite 'paresseuse' afin de limiter le nombre de messages nécessaires à la maintenance du système d'adressage lors de la connexion ou de la déconnexion de pairs. Pour étudier la faisabilité ainsi que la viabilité de nos propositions, nous effectuons une évaluation des performances.

Mots-clés

Grille de Données, Sources de Données Réparties à Grande Échelle, Découverte de Ressources, Évaluation de Performances.

Résumé en anglais

Nowadays, data grid applications look to share a huge number of data sources in an unstable environment where a data source may join or leave the system at any time. These data sources are highly heterogeneous because they are independently developed and managed and geographically scattered. In this environment, efficient discovery of relevant data sources for query execution is a complex problem due to the source heterogeneity, large scale environment and system instability. First works on data source discovery are based on a keyword search. These initial solutions are not sufficient because they do not take into account problem of semantic heterogeneity of data sources. Thus, the community has proposed other solutions to consider semantic aspects. A first solution consists in using a global schema or global ontology. However, the conception of such scheme or such ontology is a complex task due to the number of data sources. Other solutions have been proposed providing mappings between data source schemas or based on domain ontologies and establishing mapping relations between them. All these solutions impose a fixed topology for connections as well as mapping relationships. However, the definition of mapping relations between domain ontologies is a difficult task and imposing a fixed topology is a major inconvenience. In this perspective, we propose in this thesis a method for discovering data sources taking into account semantic heterogeneity problems in unstable and large scale environment. For that purpose, we associate a Virtual Organisation (VO) and a domain ontology to each domain and we rely on relationship mappings between existing ontologies. We do not impose any hypothesis on the relationship mapping topology, except that they form connected graph. We define an addressing system for permanent access from any OVi to another OVj despite peers' dynamicity (with $i \neq j$). We also present a method of maintenance called 'lazy' to limit the number of messages required to maintain the addressing system during the connection or disconnection of peers. To study the feasibility as well as the viability of our proposals, we make a performance evaluation.

Mots-clés en anglais

Data Grid, Data Sources Distributed in Large Scale, Resource Discovery, Performance Evaluation.